Búsqueda Binaria y Ventanas Deslizantes

Facundo Gutiérrez

FCEN - UBA

Training Camp 2025

Gracias Sponsors!

Organizador

Diamond Plus

Gracias Sponsors!

Platino

FOLDER IT

INTERNATIONAL
SOFTWARE COMPANY

Gold

NeuralSoft

Oro

JS JERÁRQUICOS

Gracias Sponsors!

Aliado

Esquema General

Esquema General

Buscar un elemento en un arreglo ordenado

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Ejemplos:

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Ejemplos:

- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 6$
  La respuesta es :

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Ejemplos:

- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 6$
  La respuesta es : "El elemento $x = 6$ se encuentra en el arreglo A por primera vez en la posición 2"

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Ejemplos:

- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 6$
  La respuesta es : "El elemento $x = 6$ se encuentra en el arreglo A por primera vez en la posición 2"
- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 7$
  La respuesta es :

Buscar un elemento en un arreglo ordenado

Problema:

Dado un arreglo ordenado de n números (A), queremos saber si el elemento x se encuentra en él. En caso de encontrarse, buscamos la posición exacta de su primera aparición.

Ejemplos:

- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 6$
  La respuesta es : "El elemento $x = 6$ se encuentra en el arreglo A por primera vez en la posición 2"
- $A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$ y buscamos el elemento $x = 7$
  La respuesta es : "El elemento $x = 7$ no se encuentra en el arreglo A"

Posibles Soluciones

Posibles Soluciones

1. **Búsqueda lineal**:

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = \overset{0}{[1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[0]$=6?

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \underset{\uparrow}{\overset{1}{3}}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[1]$=6?

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[2]$=6?

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[2]$=6?

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[2]$=6?

   - **Complejidad**: ...

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[2]$=6?

   - **Complejidad**: $\mathcal{O}(n)$

Posibles Soluciones

1. **Búsqueda lineal**: Recorro el arreglo de izquierda a derecha, desde $i = 0$ hasta $i = n - 1$, y en cada posición pregunto: "¿$A[i] = x$?"

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}]$$

¿$A[2]$=6?

   - **Complejidad**: $\mathcal{O}(n)$
   - Si en algún momento **encuentro una aparición de** x puedo reportarlo y terminar. Si para todo valor de i de 0 a $n - 1$ tenemos que $A[i] \neq x$, entonces reportamos que x **no se encuentra en el arreglo**.

Ahora, ¿qué información no estamos utilizando del arreglo?

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \underbrace{\overset{8}{13}}_{¿A[8]<x?}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \underbrace{\overset{3}{6}}_{\text{¿}A[3]<x\text{?}}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = \overset{0}{[1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \underset{\text{¿}A[5]<x\text{?}}{\overset{5}{8}}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = \overset{0}{[1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \underset{\text{¿}A[6]<x\text{?}}{\overset{6}{10}}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \underset{\text{¿}A[7]<x\text{?}}{\overset{7}{11}}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\overset{0}{1}, \overset{1}{3}, \overset{2}{6}, \overset{3}{6}, \overset{4}{8}, \overset{5}{8}, \overset{6}{10}, \overset{7}{11}, \overset{8}{13}, \overset{9}{14}, \overset{10}{15}, \overset{11}{15}, \overset{12}{17}, \overset{13}{18}, \overset{14}{18}, \overset{15}{20}, \overset{16}{21}]$$

Ahora, ¿qué información no estamos utilizando del arreglo?

2. **Búsqueda Binaria**: Usando que el arreglo está ordenado, podemos aprovecharlo para encontrar el índice de una manera más rápida (en una menor cantidad de "pasos" suponiendo el peor caso).
   La idea será **mantener tres rangos**. Uno con los números que sabemos que son menores, otro con los que sabemos que son mayores o iguales y otro donde **no sabemos cómo se comparan** con x. La idea es ir reduciendo el tamaño de este último rango a **la mitad en cada paso**. ¿Cómo podemos hacerlo?

Ejemplo: Buscamos el elemento $x = 11$

$$A = [\underbrace{1, 3, 6, 6, 8, 8, 10}_{A[i]<x}, \underbrace{11, 13, 14, 15, 15, 17, 18, 18, 20, 21}_{A[i]\geq x}]$$

(índices: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16; ↓ en 6 y 7)

Código

```
int busquedaBinaria (int x, vector<int> &A)
{
    int n = int(A.size());
    int a = -1, b = n; // de a para atras <
    while (b-a > 1)    // de b para adelante >=
    { // [a+1...b-1] : RANGO ACTIVO
        int medio = (a+b)/2; // punto medio de [a..b]
        if (A[medio] < x)
                a = medio;
        else
                b = medio;
    }
    if (b < n && A[b] == x) // a = 6, b = 7
        return b;           // en el ejemplo anterior
    else
        return -1;
}
```

Código

```
int busquedaBinaria (int x, vector<int> &A)
{
    int n = int(A.size());
    int a = -1, b = n; // de a para atras <
    while (b-a > 1)    // de b para adelante >=
    { // [a+1...b-1] : RANGO ACTIVO
        int medio = (a+b)/2; // punto medio de [a..b]
        if (A[medio] < x)
                a = medio;
        else
                b = medio;
    }
    if (b < n && A[b] == x) // a = 6, b = 7
        return b;           // en el ejemplo anterior
    else
        return -1;
}
```

Complejidad : ...

Código

```
int busquedaBinaria (int x, vector<int> &A)
{
    int n = int(A.size());
    int a = -1, b = n; // de a para atras <
    while (b-a > 1)    // de b para adelante >=
    { // [a+1...b-1] : RANGO ACTIVO
        int medio = (a+b)/2; // punto medio de [a..b]
        if (A[medio] < x)
                a = medio;
        else
                b = medio;
    }
    if (b < n && A[b] == x) // a = 6, b = 7
        return b;           // en el ejemplo anterior
    else
        return -1;
}
```

Complejidad : $\mathcal{O}(\lg(n))$

Ahora vamos a utilizar esta **misma idea para resolver más problemas**, y no solo : "Cómo buscar un número en un arreglo ordenado".

Ahora vamos a utilizar esta **misma idea para resolver más problemas**, y no solo : "Cómo buscar un número en un arreglo ordenado".

Supongamos que tenemos una cierta **propiedad que depende de un valor**. Llamémosla $P(k)$. Por ejemplo: $P(k) : k^2 > 29$. Nos queda que para un cierto k, $P(k)$ puede ser Verdadera o Falsa.

Ahora vamos a utilizar esta **misma idea para resolver más problemas**, y no solo : "Cómo buscar un número en un arreglo ordenado".

Supongamos que tenemos una cierta **propiedad que depende de un valor**. Llamémosla $P(k)$. Por ejemplo: $P(k) : k^2 > 29$. Nos queda que para un cierto k, $P(k)$ puede ser Verdadera o Falsa.

- $P(1) : 1^2 > 29 \iff 1 > 29 \rightarrow \textit{Falsa}$
- $P(2) : 2^2 > 29 \iff 4 > 29 \rightarrow \textit{Falsa}$
- $P(3) : 3^2 > 29 \iff 9 > 29 \rightarrow \textit{Falsa}$
- $P(4) : 4^2 > 29 \iff 16 > 29 \rightarrow \textit{Falsa}$
- $P(5) : 5^2 > 29 \iff 25 > 29 \rightarrow \textit{Falsa}$
- $P(6) : 6^2 > 29 \iff 36 > 29 \rightarrow \textit{Verdadera}$
- $P(7) : 7^2 > 29 \iff 49 > 29 \rightarrow \textit{Verdadera}$
- $P(8) : 8^2 > 29 \iff 64 > 29 \rightarrow \textit{Verdadera}$

Si nuestra propiedad cumple que es Verdadera hasta un cierto punto y luego es siempre Falsa (o al revés, Falsa hasta un cierto punto y luego Verdadera), decimos que es una **propiedad binaria**.

Si nuestra propiedad cumple que es Verdadera hasta un cierto punto y luego es siempre Falsa (o al revés, Falsa hasta un cierto punto y luego Verdadera), decimos que es una **propiedad binaria**. En estos casos, podemos hacer **búsqueda binaria**, para encontrar el **momento en el que cambia de valor** nuestra propiedad.

Si nuestra propiedad cumple que es Verdadera hasta un cierto punto y luego es siempre Falsa (o al revés, Falsa hasta un cierto punto y luego Verdadera), decimos que es una **propiedad binaria**. En estos casos, podemos hacer **búsqueda binaria**, para encontrar el **momento en el que cambia de valor** nuestra propiedad.

k	0	1	...	a	b	b+1	...	n
$P(k)$	Falsa	Falsa	Falsa	Falsa	Verdad	Verdad	Verdad	Verdad

Si nuestra propiedad cumple que es Verdadera hasta un cierto punto y luego es siempre Falsa (o al revés, Falsa hasta un cierto punto y luego Verdadera), decimos que es una **propiedad binaria**. En estos casos, podemos hacer **búsqueda binaria**, para encontrar el **momento en el que cambia de valor** nuestra propiedad.

k	0	1	...	a	b	b+1	...	n
$P(k)$	Falsa	Falsa	Falsa	Falsa	Verdad	Verdad	Verdad	Verdad

Búsqueda binaria nos permite encontrar el **primer momento** en que nuestra propiedad resulta ser **verdadera** así como el **último momento** en que nuestra propiedad es **falsa** (O al revés, dependiendo de qué propiedad planteamos).

Código

Código

¿Es muy distinto el código al anterior?

Código

¿Es muy distinto el código al anterior? NO

```
void busquedaBinaria (int &a, int &b)
{                                       // P(a) : Falso
    while (b-a > 1)                     // P(b) : Verdadero
    { // [a+1...b-1] : RANGO ACTIVO
        int medio = (a+b)/2; // punto medio de [a..b]
        if (P(medio))
                b = medio;
        else
                a = medio;
    }
    // Ahora en a queda el ultimo que es Falso
    // Y en b queda el primero que es Verdadero
}
```

Evolucionando Pokémones

Ash acaba de atrapar N **pokémones**. Él tiene a su disposición M **barras de caramelo**. Evolucionar a un pokémon, le cuesta X barras de caramelo. A su vez, puede vender a un pokémon y recibir a cambio Y barras de caramelo. ¿Cuál es la máxima cantidad de pokémones que puede evolucionar?

Solución

Solución

Observación clave

Si **fijamos la cantidad de pokémones que vamos a evolucionar**, digamos k con $0 \leq k \leq N$. Entonces podemos vender a los $N - k$ pokemones que no vamos a evolucionar. Con esto, disponemos de $M + (N - k) \cdot Y$ barras de caramelo. Para evolucionar a los k pokemones, necesitamos tener al menos $k \cdot X$ barras de caramelos. O sea, debe valer que:

$$k \cdot X \leq (N - k) \cdot Y + M$$

Solución

Observación clave

Si **fijamos la cantidad de pokémones que vamos a evolucionar**, digamos k con $0 \leq k \leq N$. Entonces podemos vender a los $N - k$ pokemones que no vamos a evolucionar. Con esto, disponemos de $M + (N - k) \cdot Y$ barras de caramelo. Para evolucionar a los k pokemones, necesitamos tener al menos $k \cdot X$ barras de caramelos. O sea, debe valer que:

$$k \cdot X \leq (N - k) \cdot Y + M$$

1. Solución: Probar todos los valores de k entre 0 y N (inclusive), y tomar el de mayor valor que satisfaga la condición. Complejidad : $\mathcal{O}(N)$

Solución

Solución

2. Solución: Notemos que **si podemos evolucionar** k **pokemones, también podemos evolucionar** $k-1$**,** $k-2$**,** $\ldots$**,** 0 **pokemones**. En particular siempre podemos evolucionar 0 pokemones.
   De la misma forma, **si no podemos evolucionar** k **pokemones, menos aún podremos evolucionar** $k+1$**, o** $k+2$ **pokemones** (o cualquier otro número mayor). En particular nunca podremos evolucionar $N+1$ pokemones.

Solución

2 Solución: Notemos que **si podemos evolucionar** k **pokemones, también podemos evolucionar** $k-1$**,** $k-2$**,** $\ldots$**,** 0 **pokemones**. En particular siempre podemos evolucionar 0 pokemones.
De la misma forma, **si no podemos evolucionar** k **pokemones, menos aún podremos evolucionar** $k+1$**, o** $k+2$ **pokemones** (o cualquier otro número mayor). En particular nunca podremos evolucionar $N+1$ pokemones.
Entonces, tomando como propiedad:
$P(k)$: "Puedo evolucionar k pokemones", **podemos hacer búsqueda binaria en la propiedad** P.

Solución

2. Solución: Notemos que **si podemos evolucionar** k **pokemones, también podemos evolucionar** $k-1$**,** $k-2$**,** $\ldots$**,** 0 **pokemones**. En particular siempre podemos evolucionar 0 pokemones.
   De la misma forma, **si no podemos evolucionar** k **pokemones, menos aún podremos evolucionar** $k+1$**, o** $k+2$ **pokemones** (o cualquier otro número mayor). En particular nunca podremos evolucionar $N+1$ pokemones.
   Entonces, tomando como propiedad:
   $P(k)$: "Puedo evolucionar k pokemones", **podemos hacer búsqueda binaria en la propiedad** P.
   Complejidad: $\mathcal{O}(\lg(N))$

Solución

Solución

3. Sabemos que $0 \leq k \leq N$. Y de la condición vista, podemos despejar una inecuación para k:

Solución

3. Sabemos que $0 \leq k \leq N$. Y de la condición vista, podemos despejar una inecuación para k:

$$\begin{aligned} k \cdot X &\leq (N - k) \cdot Y + M && \iff \\ k \cdot X &\leq N \cdot Y - k \cdot Y + M && \iff \\ k \cdot X + k \cdot Y &\leq N \cdot Y + M && \iff \\ k \cdot (X + Y) &\leq N \cdot Y + M && \iff \\ k &\leq \frac{N \cdot Y + M}{X + Y} \end{aligned}$$

Solución

3. Sabemos que $0 \leq k \leq N$. Y de la condición vista, podemos despejar una inecuación para k:

$$\begin{aligned} k \cdot X &\leq (N-k) \cdot Y + M && \iff \\ k \cdot X &\leq N \cdot Y - k \cdot Y + M && \iff \\ k \cdot X + k \cdot Y &\leq N \cdot Y + M && \iff \\ k \cdot (X+Y) &\leq N \cdot Y + M && \iff \\ k &\leq \frac{N \cdot Y + M}{X+Y} \end{aligned}$$

Por lo tanto, la respuesta será :

$$\boxed{\min\left(N, \left\lfloor \frac{N \cdot Y + M}{X+Y} \right\rfloor\right)}$$

Solución

3. Sabemos que $0 \leq k \leq N$. Y de la condición vista, podemos despejar una inecuación para k:

$$\begin{aligned} k \cdot X &\leq (N-k) \cdot Y + M && \iff \\ k \cdot X &\leq N \cdot Y - k \cdot Y + M && \iff \\ k \cdot X + k \cdot Y &\leq N \cdot Y + M && \iff \\ k \cdot (X+Y) &\leq N \cdot Y + M && \iff \\ k &\leq \frac{N \cdot Y + M}{X+Y} \end{aligned}$$

Por lo tanto, la respuesta será :

$$\boxed{\min\left(N, \left\lfloor \frac{N \cdot Y + M}{X+Y} \right\rfloor\right)}$$

Complejidad: $\mathcal{O}(1)$

Solución

3. Sabemos que $0 \leq k \leq N$. Y de la condición vista, podemos despejar una inecuación para k:

$$\begin{aligned} k \cdot X &\leq (N-k) \cdot Y + M && \iff \\ k \cdot X &\leq N \cdot Y - k \cdot Y + M && \iff \\ k \cdot X + k \cdot Y &\leq N \cdot Y + M && \iff \\ k \cdot (X+Y) &\leq N \cdot Y + M && \iff \\ k &\leq \frac{N \cdot Y + M}{X+Y} \end{aligned}$$

Por lo tanto, la respuesta será :

$$\boxed{\min\left(N, \left\lfloor \frac{N \cdot Y + M}{X+Y} \right\rfloor\right)}$$

Complejidad: $\mathcal{O}(1)$

https://csacademy.com/contest/archive/task/pokemon-evolution

¿Preguntas?

Problema de Optimización $\rightarrow$ Problema de Decisión

Problema de Optimización $\rightarrow$ Problema de Decisión

En este problema que vimos aparece un patrón que es común en los problemas que se resuelven con búsqueda binaria.

Lo que hicimos fue **convertir un problema de optimización** (¿cuál es la máxima cantidad de pokemones que podemos evolucionar?), **en uno de decisión (Sí/No)** (¿Puedo evolucionar k pokemones?).

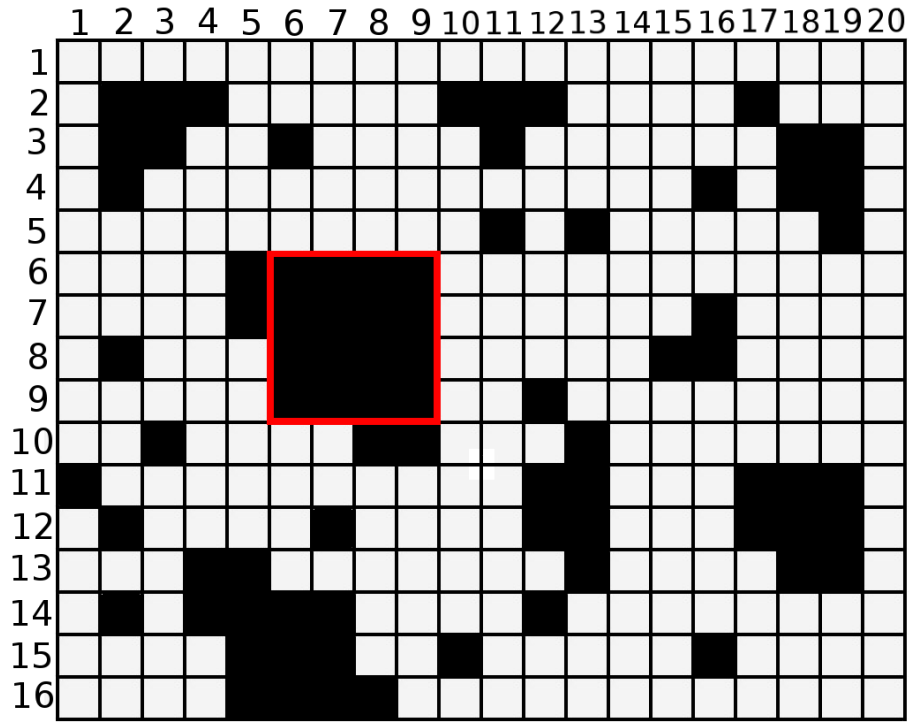
Veamos más ejemplos.

Máximo Cuadrado en un Tablero

Tenemos un tablero de $m \times n$ (m filas y n columnas). Algunas casillas del tablero están pintadas de blanco y otras de negro. Buscamos encontrar el tamaño del lado del subtablero cuadrado más grande de casillas negras.

Máximo Cuadrado en un Tablero

Tenemos un tablero de $m \times n$ (m filas y n columnas). Algunas casillas del tablero están pintadas de blanco y otras de negro. Buscamos encontrar el tamaño del lado del subtablero cuadrado más grande de casillas negras.



Solución

Solución

Cosas que vamos a usar

Vamos a suponer que tenemos una función booleana $f(k, i, j)$ que **nos dice si hay un cuadrado de casillas negras de lado** k en el tablero cuya casilla superior izquierda está situada en la fila i y la columna j. Por ejemplo, en la imagen anterior, $f(2, 2, 2)$ es `true` y $f(3, 2, 2)$ es `false`.

Solución

Cosas que vamos a usar

Vamos a suponer que tenemos una función booleana $f(k, i, j)$ que **nos dice si hay un cuadrado de casillas negras de lado** k en el tablero cuya casilla superior izquierda está situada en la fila i y la columna j. Por ejemplo, en la imagen anterior, $f(2, 2, 2)$ es `true` y $f(3, 2, 2)$ es `false`.

1. Solución: El tamaño del cuadrado que buscamos es **a lo sumo tan grande como la dimensión más chica del tablero**. Podemos **recorrer todas las casillas**, y en cada una preguntarle a la función f si existe un tablero de tamaño $0, 1, 2, \ldots, \min(n, m)$ inclusive. El valor más grande que devolvió `true` será el lado del cuadrado más grande que tiene la esquina superior izquierda en la casilla que estamos analizando. Tomando el máximo de este número para todas las casillas resolvemos el problema.

Solución

Cosas que vamos a usar

Vamos a suponer que tenemos una función booleana $f(k, i, j)$ que **nos dice si hay un cuadrado de casillas negras de lado** k en el tablero cuya casilla superior izquierda está situada en la fila i y la columna j. Por ejemplo, en la imagen anterior, $f(2, 2, 2)$ es `true` y $f(3, 2, 2)$ es `false`.

1. Solución: El tamaño del cuadrado que buscamos es **a lo sumo tan grande como la dimensión más chica del tablero**. Podemos **recorrer todas las casillas**, y en cada una preguntarle a la función f si existe un tablero de tamaño $0, 1, 2, \ldots, \min(n, m)$ inclusive. El valor más grande que devolvió `true` será el lado del cuadrado más grande que tiene la esquina superior izquierda en la casilla que estamos analizando. Tomando el máximo de este número para todas las casillas resolvemos el problema. ¿Cuántas veces estamos llamando a la función f?

Solución

Cosas que vamos a usar

Vamos a suponer que tenemos una función booleana $f(k, i, j)$ que **nos dice si hay un cuadrado de casillas negras de lado** k en el tablero cuya casilla superior izquierda está situada en la fila i y la columna j. Por ejemplo, en la imagen anterior, $f(2, 2, 2)$ es `true` y $f(3, 2, 2)$ es `false`.

1. Solución: El tamaño del cuadrado que buscamos es **a lo sumo tan grande como la dimensión más chica del tablero**. Podemos **recorrer todas las casillas**, y en cada una preguntarle a la función f si existe un tablero de tamaño $0, 1, 2, \ldots, \min(n, m)$ inclusive. El valor más grande que devolvió `true` será el lado del cuadrado más grande que tiene la esquina superior izquierda en la casilla que estamos analizando. Tomando el máximo de este número para todas las casillas resolvemos el problema. ¿Cuántas veces estamos llamando a la función f?

   Respuesta : $\boxed{n \cdot m \cdot \min(n, m) \sim n^3}$

Solución

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Búsqueda binaria

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)
Búsqueda binaria Ok... ¿Pero en qué?

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Búsqueda binaria Ok... ¿Pero en qué?

2. En la solución anterior, cuando estamos parados en una casilla, podemos hacer búsqueda binaria en **el lado del cuadrado que buscamos**. ¿Por qué?

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Búsqueda binaria Ok... ¿Pero en qué?

2. En la solución anterior, cuando estamos parados en una casilla, podemos hacer búsqueda binaria en **el lado del cuadrado que buscamos**. ¿Por qué?
   Supongamos que desde una casilla hay un subtablero de tamaño $k \times k$, entonces desde esa misma casilla es claro que también hay subtableros de lado $k-1$, $k-2$, etc. Por otro lado, si no hay un subtablero de lado k, tampoco habrá uno de lado $k+1$ o más. (hacer siempre dibujitos).

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Búsqueda binaria Ok... ¿Pero en qué?

2. En la solución anterior, cuando estamos parados en una casilla, podemos hacer búsqueda binaria en **el lado del cuadrado que buscamos**. ¿Por qué?
   Supongamos que desde una casilla hay un subtablero de tamaño $k \times k$, entonces desde esa misma casilla es claro que también hay subtableros de lado $k-1$, $k-2$, etc. Por otro lado, si no hay un subtablero de lado k, tampoco habrá uno de lado $k+1$ o más. (hacer siempre dibujitos).

¿Cuántas veces estamos llamando a la función f?

Solución

¿Cómo podemos **acelerar la solución**? (hacer menos llamados a la f)

Búsqueda binaria Ok... ¿Pero en qué?

2. En la solución anterior, cuando estamos parados en una casilla, podemos hacer búsqueda binaria en **el lado del cuadrado que buscamos**. ¿Por qué?
   Supongamos que desde una casilla hay un subtablero de tamaño $k \times k$, entonces desde esa misma casilla es claro que también hay subtableros de lado $k-1$, $k-2$, etc. Por otro lado, si no hay un subtablero de lado k, tampoco habrá uno de lado $k+1$ o más. (hacer siempre dibujitos).

¿Cuántas veces estamos llamando a la función f?

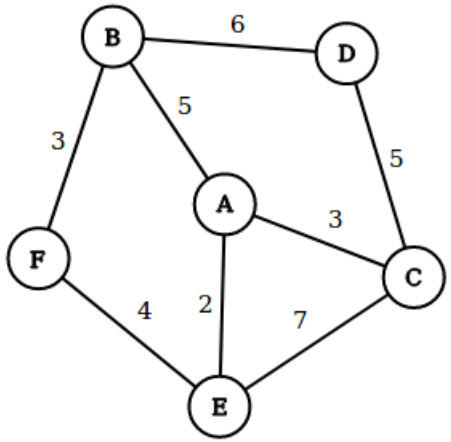
Respuesta : $\boxed{n \cdot m \cdot \lg(\min(n,m)) \sim n^2 \cdot \lg(n)}$

Código

```
int cuadradoEnTablero (vector<vector<int> > &tablero)
{
  int respuesta = 0; // siempre hay cuadrado de lado 0.
  int m = int(tablero.size()), n = int(tablero[0].size()); // filas
  for (int i = 0; i < m; i++)                              // y columnas
  for (int j = 0; j < n; j++)
  {
    int a = 0, b = min(n,m)+1; // Hay cuadrado de lado 0,
    while (b-a > 1)            // No hay cuadrado de lado min(n,m)+1
    {
      int k = (a+b)/2;
      if (f(k,i,j,tablero))
        a = k;
      else
        b = k;
    } // En a queda guardado el lado del cuadrado
      // mas grande que existe desde la casilla (i,j)
    respuesta = max(respuesta,a);
  }
  return respuesta;
}
```

Problema

Petr vive en un pueblo de Rusia. Las rutas de su país tienen un "índice de peligrosidad". Petr quiere viajar del pueblo A (donde él vive) a B (donde vive su mejor amigo), **minimizando** el índice de peligrosidad de **la ruta más peligrosa que usa en su camino**.



Por ejemplo, en el pueblo de la figura, lo mejor es tomar el camino $A \xrightarrow{2} E \xrightarrow{4} F \xrightarrow{3} B$, cuya ruta más peligrosa es de 4.

Solución

Solución

Si bien hay muchas formas de resolver el problema, vamos a hacerlo con **Búsqueda Binaria**

Solución

Si bien hay muchas formas de resolver el problema, vamos a hacerlo con **Búsqueda Binaria**
Si **fijamos el valor de la ruta más peligrosa a utilizar**, digamos que la ruta más peligrosa que usamos tiene índice de peligrosidad x (y descartamos todas las rutas con peligrosidad mayor a x).
Entonces si en este nuevo diseño de país hay un camino (cualquiera) de A a B, sabemos que existe un camino cuya ruta más peligrosa es a lo sumo x.

Solución

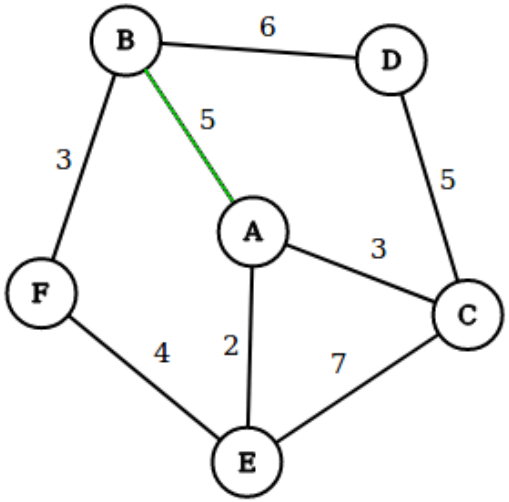
Si bien hay muchas formas de resolver el problema, vamos a hacerlo con **Búsqueda Binaria**
Si **fijamos el valor de la ruta más peligrosa a utilizar**, digamos que la ruta más peligrosa que usamos tiene índice de peligrosidad x (y descartamos todas las rutas con peligrosidad mayor a x).
Entonces si en este nuevo diseño de país hay un camino (cualquiera) de A a B, sabemos que existe un camino cuya ruta más peligrosa es a lo sumo x.
Si consideramos la propiedad $P(x)$: "Hay un camino entre A y B cuya ruta más peligrosa es menor o igual a x ", entonces esta **propiedad es binaria**. ¿Por qué?

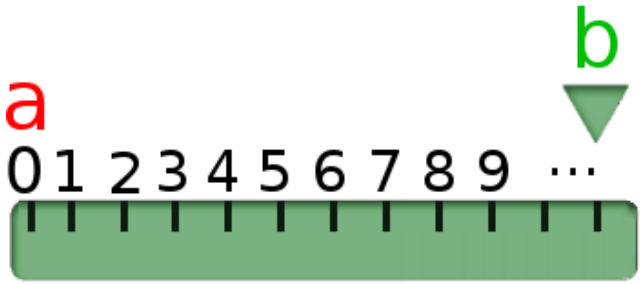
Solución

Si bien hay muchas formas de resolver el problema, vamos a hacerlo con **Búsqueda Binaria**
Si **fijamos el valor de la ruta más peligrosa a utilizar**, digamos que la ruta más peligrosa que usamos tiene índice de peligrosidad x (y descartamos todas las rutas con peligrosidad mayor a x).
Entonces si en este nuevo diseño de país hay un camino (cualquiera) de A a B, sabemos que existe un camino cuya ruta más peligrosa es a lo sumo x.
Si consideramos la propiedad $P(x)$: "Hay un camino entre A y B cuya ruta más peligrosa es menor o igual a x ", entonces esta **propiedad es binaria**. ¿Por qué?
Si tenemos un camino cuya ruta más peligrosa tiene peligrosidad menor o igual a x, entonces tenemos un camino cuya ruta más peligrosa es menor o igual a $x+1$, ... (el mismo camino funciona).

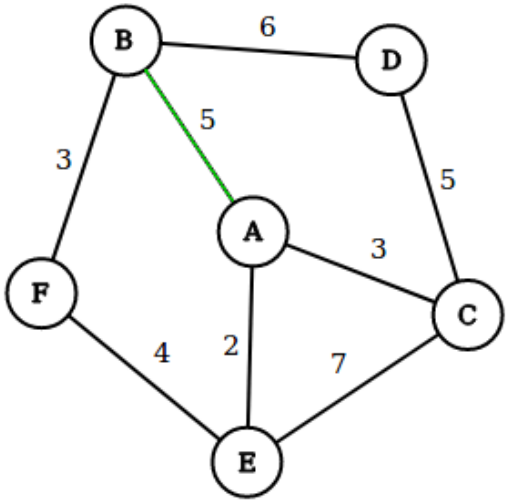
Ejemplo gráfico

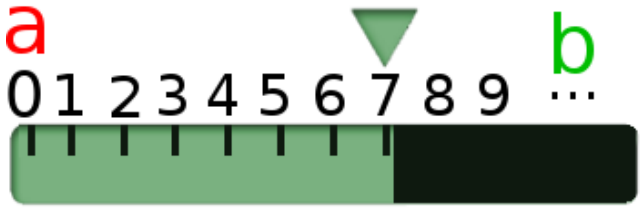
Ejemplo gráfico



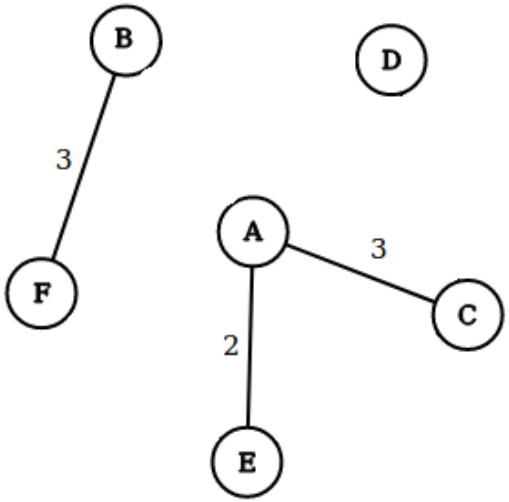


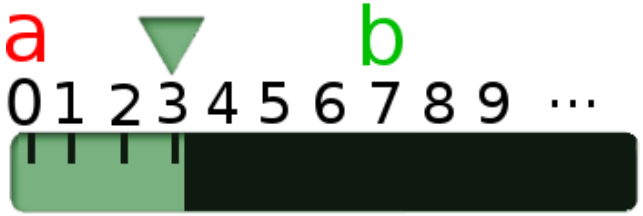
Ejemplo gráfico



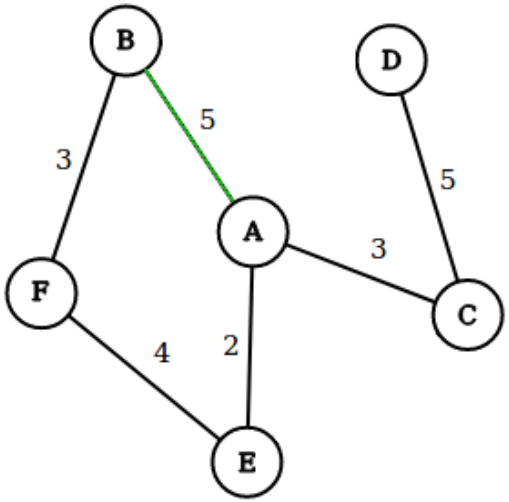


Ejemplo gráfico

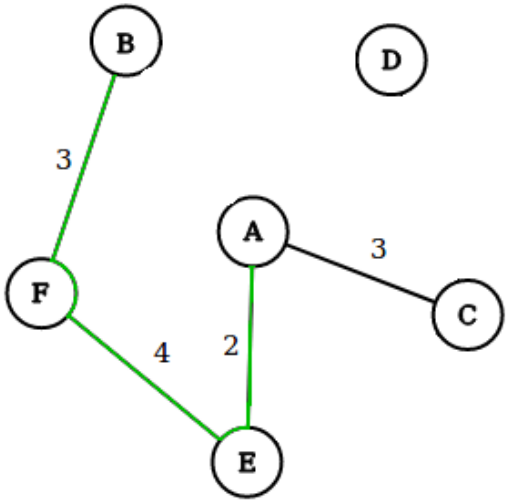




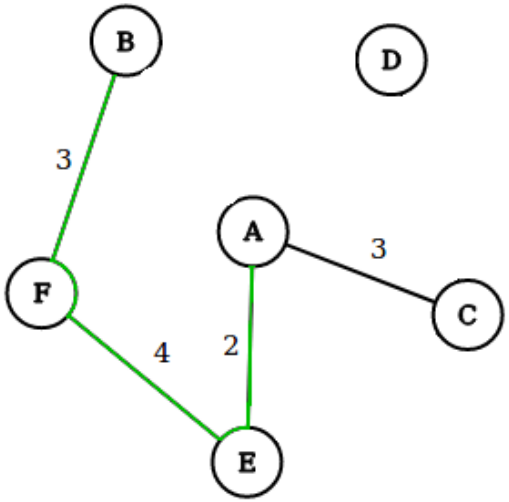
Ejemplo gráfico



Ejemplo gráfico



Ejemplo gráfico



¿Preguntas?

Esquema General

Ventanas Deslizantes

Ventanas Deslizantes

Problema:

Dado un arreglo de n números **positivos**, y un número x, nos interesa saber si existe un subarreglo cuya suma sea x.

Ventanas Deslizantes

Problema:

Dado un arreglo de n números **positivos**, y un número x, nos interesa saber si existe un subarreglo cuya suma sea x.

Ejemplos:

- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 8$
  La respuesta es :

Ventanas Deslizantes

Problema:

Dado un arreglo de n números **positivos**, y un número x, nos interesa saber si existe un subarreglo cuya suma sea x.

Ejemplos:

- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 8$
  La respuesta es : "El subarreglo $[2..5)$ suma 8"

Ventanas Deslizantes

Problema:

Dado un arreglo de n números **positivos**, y un número x, nos interesa saber si existe un subarreglo cuya suma sea x.

Ejemplos:

- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 8$
  La respuesta es : "El subarreglo $[2..5)$ suma 8"
- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 4$
  La respuesta es :

Ventanas Deslizantes

Problema:

Dado un arreglo de n números **positivos**, y un número x, nos interesa saber si existe un subarreglo cuya suma sea x.

Ejemplos:

- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 8$
  La respuesta es : "El subarreglo $[2..5)$ suma 8"
- $A = [\overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$. Buscamos sumar $x = 4$
  La respuesta es : "No hay subarreglo de A que sume 4"

Solución

Solución

1. **Probar todos los subarreglos posibles**. Es decir, todos los subarreglos $[i..j)$ con $0 \leq j \leq n+1$ y $0 \leq i < j$. Para cada uno de ellos calcular `suma`(i,j), la suma de los números en el subarreglo especificado y ver si es exactamente x.
   Complejidad :

Solución

1. **Probar todos los subarreglos posibles**. Es decir, todos los subarreglos $[i..j)$ con $0 \leq j \leq n+1$ y $0 \leq i < j$. Para cada uno de ellos calcular `suma(i,j)`, la suma de los números en el subarreglo especificado y ver si es exactamente x.
   Complejidad : $\mathcal{O}(n^3)$ u $\mathcal{O}(n^2)$ dependiendo de la implementación de la función `suma`.

Solución

1. **Probar todos los subarreglos posibles**. Es decir, todos los subarreglos $[i..j)$ con $0 \leq j \leq n+1$ y $0 \leq i < j$. Para cada uno de ellos calcular `suma`(i,j), la suma de los números en el subarreglo especificado y ver si es exactamente x.
   Complejidad : $\mathcal{O}(n^3)$ u $\mathcal{O}(n^2)$ dependiendo de la implementación de la función `suma`.
2. Utilizar una **ventana deslizante** para resolver el problema. La idea es mantener **dos índices** que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos "i" y al derecho lo llamaremos "j". Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a **mantener** cuánto vale **la suma de los números dentro de la ventana [i..j)** (notar que no incluimos el extremo derecho).

2. Utilizar una **ventana deslizante** para resolver el problema. La idea es mantener **dos índices** que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos "*i*" y al derecho lo llamaremos "*j*". Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a **mantener** cuánto vale **la suma de los números dentro de la ventana [i..j)** (notar que no incluimos el extremo derecho).

2. Utilizar una **ventana deslizante** para resolver el problema. La idea es mantener **dos índices** que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos "i" y al derecho lo llamaremos "j". Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a **mantener** cuánto vale **la suma de los números dentro de la ventana [i..j)** (notar que no incluimos el extremo derecho).

Ejemplo: Si $i = 1$ y $j = 4$, mantenemos la suma en el subarreglo $[i..j) = [1,4) = [1..3]$.

$$A = \overset{0}{[2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}]$$

2. Utilizar una **ventana deslizante** para resolver el problema. La idea es mantener **dos índices** que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos "i" y al derecho lo llamaremos "j". Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a **mantener** cuánto vale **la suma de los números dentro de la ventana [i..j)** (notar que no incluimos el extremo derecho).

Ejemplo: Si $i = 1$ y $j = 4$, mantenemos la suma en el subarreglo $[i..j) = [1,4) = [1..3]$.

$$A = [2, \underbrace{3, 2, 5}_{\texttt{suma=10}}, 1, 5, 2, 3]$$

$$\begin{matrix} & & i & & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$

Entrada

Buscamos sumar $x = 8$

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$

Paso 1

Buscamos sumar $x = 8$

$$\texttt{suma} = 0 < x$$

$$A = \begin{matrix} & & & & ij & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ [2, 3, 2, 5, 1, 5, 2, 3]$$

Paso 1

Buscamos sumar $x = 8$

$\texttt{suma} = 0 < x \rightarrow$ Aumento j

$$\begin{array}{c} \\ A = \end{array} \begin{array}{c} \begin{matrix} i & & & & & & & j \end{matrix} \\ \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ [2, 3, 2, 5, 1, 5, 2, 3] \end{array}$$

Paso 2

Buscamos sumar $x = 8$

$$\texttt{suma} = 2 < x$$

$$A = [\underbrace{\begin{matrix} i \\ 0 \\ 2 \end{matrix}}, \begin{matrix} j \\ 1 \\ 3 \end{matrix}, \begin{matrix} 2 \\ 2 \end{matrix}, \begin{matrix} 3 \\ 5 \end{matrix}, \begin{matrix} 4 \\ 1 \end{matrix}, \begin{matrix} 5 \\ 5 \end{matrix}, \begin{matrix} 6 \\ 2 \end{matrix}, \begin{matrix} 7 \\ 3 \end{matrix}]$$

Paso 2

Buscamos sumar $x = 8$

$\texttt{suma} = 2 < x \rightarrow$ Aumento j

$$A = [\underbrace{\begin{matrix} i \\ 2 \end{matrix}}_{\begin{matrix} j \\ 3, 2, 5, 1, 5, 2, 3 \end{matrix}}]$$

Paso 3

Buscamos sumar $x = 8$

$\texttt{suma} = 5 < x$

$$A = [\underbrace{2, 3}, 2, 5, 1, 5, 2, 3]$$

i en la posición 0, j en la posición 2

Paso 3

Buscamos sumar $x = 8$

$\texttt{suma} = 5 < x \rightarrow$ Aumento j

$$A = [\underbrace{2, 3}, 2, 5, 1, 5, 2, 3]$$

i en 0, j en 2

Paso 4

Buscamos sumar $x = 8$

$$\texttt{suma} = 7 < x$$

$$A = [\underbrace{2, 3, 2}, 5, 1, 5, 2, 3]$$

i en el índice 0, j en el índice 3

Paso 4

Buscamos sumar $x = 8$

$\texttt{suma} = 7 < x \rightarrow$ Aumento j

$$A = [\underbrace{2, 3, 2}, 5, 1, 5, 2, 3]$$

i			j				
0	1	2	3	4	5	6	7

Paso 5

Buscamos sumar $x = 8$

$$\texttt{suma} = 12 > x$$

$$\begin{array}{c} \overset{i}{0}\ \ 1\ \ 2\ \ 3\ \ \overset{j}{4}\ \ 5\ \ 6\ \ 7 \\ A = [\underbrace{2, 3, 2, 5}, 1, 5, 2, 3] \end{array}$$

Paso 5

Buscamos sumar $x = 8$

$\texttt{suma} = 12 > x \rightarrow$ Aumento i

$$A = [\underbrace{2, 3, 2, 5}, 1, 5, 2, 3]$$

i en la posición 0, j en la posición 4

Paso 6

Buscamos sumar $x = 8$

$$\texttt{suma} = 10 > x$$

$$A = [2, \underbrace{3, 2, 5}, 1, 5, 2, 3]$$

$$\begin{matrix} & & i & & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$

Paso 6

Buscamos sumar $x = 8$

$$\texttt{suma} = 10 > x \rightarrow \text{Aumento } i$$

$$A = [2, \underbrace{3, 2, 5}, 1, 5, 2, 3]$$

i en la posición 1, j en la posición 4

0	1	2	3	4	5	6	7
2	3	2	5	1	5	2	3

Paso 7

Buscamos sumar $x = 8$

$\texttt{suma} = 7 < x$

$$A = [2, 3, \underbrace{2, 5}, 1, 5, 2, 3]$$

i en la posición 2, j en la posición 4

Paso 7

Buscamos sumar $x = 8$

$\texttt{suma} = 7 < x \rightarrow$ Aumento j

$$A = [2, 3, \underbrace{2, 5}_{i \ldots j}, 1, 5, 2, 3]$$

$$\begin{matrix} & & & i & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$

Paso 8

Buscamos sumar $x = 8$

$$\texttt{suma} = 8 = x$$

$$A = [2, 3, \underbrace{2, 5, 1}, 5, 2, 3]$$

i en la posición 2, j en la posición 5

0	1	2	3	4	5	6	7
2	3	2	5	1	5	2	3

Paso 8

Buscamos sumar $x = 8$

$\texttt{suma} = 8 = x \rightarrow$ Podemos terminar

$$A = [2, 3, \underbrace{2, 5, 1}, 5, 2, 3]$$

i en la posición 2, j en la posición 5

0	1	2	3	4	5	6	7

Análisis

Análisis

- ¿Cuántos pasos hace el algoritmo?
  Respuesta :

Análisis

- ¿Cuántos pasos hace el algoritmo?
  Respuesta : En cada paso, o bien **aumenta i** o **aumenta j**. Cada uno de ellos puede ser aumentado a lo sumo n veces. Por lo tanto al cabo de $2n$ **pasos** en el peor caso, finaliza nuestro algoritmo.

Análisis

- ¿Cuántos pasos hace el algoritmo?
  Respuesta : En cada paso, o bien **aumenta i** o **aumenta j**. Cada uno de ellos puede ser aumentado a lo sumo n veces. Por lo tanto al cabo de $2n$ **pasos** en el peor caso, finaliza nuestro algoritmo.
- ¿Por qué es importante que los números sean **positivos**?
  Respuesta :

Análisis

- ¿Cuántos pasos hace el algoritmo?
  Respuesta : En cada paso, o bien **aumenta i** o **aumenta j**. Cada uno de ellos puede ser aumentado a lo sumo n veces. Por lo tanto al cabo de $2n$ **pasos** en el peor caso, finaliza nuestro algoritmo.
- ¿Por qué es importante que los números sean **positivos**?
  Respuesta : Porque nos permite saber que si en algún momento $\texttt{suma} > x$, entonces todas las ventanas que tienen el mismo valor de i y un j mayor tendrán una suma mayor y podemos no mirarlas (de alguna forma, podemos afirmar que **"ya nos pasamos"**).

Código

Código

```
// En iR, jR devolvemos la respuesta
void ventanaDeslizante(vector<int> &A, int x, int &iR, int &jR)
{
    int n = int(A.size()), j = 0, suma = 0; // La suma en [0,0) es 0
    for(int i = 0; i < n; i++) // para cada ventana que empieza en i:
    {
        // Ingresamos a la ventana hasta pasarnos con la suma desde i
        while (j < n && suma < x)
        {
            suma += A[j];
            j++;
        }
        // Al salir del while: (suma >= x) o (j == n)
        if (suma == x)
        {
            iR = i;
            jR = j;
        }
        // Al avanzar i, sale de la ventana
        suma -= A[i];
    }
}
```

2SUM

2SUM

Problema:

Dado un arreglo de n números, y un número x. Queremos encontrar dos números del arreglo que sumen x, o reportar que no existe tal par.

2SUM

Problema:

Dado un arreglo de n números, y un número x. Queremos encontrar dos números del arreglo que sumen x, o reportar que no existe tal par.

Ejemplos:

- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$. Buscamos sumar $x = 12$
  La respuesta es :

2SUM

Problema:

Dado un arreglo de n números, y un número x. Queremos encontrar dos números del arreglo que sumen x, o reportar que no existe tal par.

Ejemplos:

- $A = [\overset{0}{7}, 10, 4, 1, 9, \overset{5}{5}, 9, 6]$. Buscamos sumar $x = 12$
  La respuesta es : "Sumando $A[0]$ y $A[5]$ obtenemos 12"

2SUM

Problema:

Dado un arreglo de n números, y un número x. Queremos encontrar dos números del arreglo que sumen x, o reportar que no existe tal par.

Ejemplos:

- $A = [\overset{0}{7}, 10, 4, 1, 9, \overset{5}{5}, 9, 6]$. Buscamos sumar $x = 12$
  La respuesta es : "Sumando $A[0]$ y $A[5]$ obtenemos 12"
- $A = [\overset{0}{7}, 10, 4, 1, 9, \overset{5}{5}, 9, 6]$. Buscamos sumar $x = 4$
  La respuesta es :

2SUM

Problema:

Dado un arreglo de n números, y un número x. Queremos encontrar dos números del arreglo que sumen x, o reportar que no existe tal par.

Ejemplos:

- $A = [\overset{0}{7}, 10, 4, 1, 9, \overset{5}{5}, 9, 6]$. Buscamos sumar $x = 12$
  La respuesta es : "Sumando $A[0]$ y $A[5]$ obtenemos 12"
- $A = [\overset{0}{7}, 10, 4, 1, 9, \overset{5}{5}, 9, 6]$. Buscamos sumar $x = 4$
  La respuesta es : "No hay dos elementos de A que sumen 4"

Solución

Solución

Observación clave

Es importante notar que el orden de los números en el arreglo A no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está **ordenado en forma creciente** (u ordenarlo en $\mathcal{O}(n \lg(n))$)

Solución

Observación clave

Es importante notar que el orden de los números en el arreglo A no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está **ordenado en forma creciente** (u ordenarlo en $\mathcal{O}(n \lg(n))$)

1. Probar todos los pares de índices, y ver si esos dos números suman o no x. De existir un par que sí, reportarlo.
   Complejidad $\mathcal{O}(n^2)$

Solución

Observación clave

Es importante notar que el orden de los números en el arreglo A no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está **ordenado en forma creciente** (u ordenarlo en $\mathcal{O}(n \lg(n))$)

1. Probar todos los pares de índices, y ver si esos dos números suman o no x. De existir un par que sí, reportarlo. Complejidad $\mathcal{O}(n^2)$
2. Utilizar una variante de la técnica que vimos recién para resolver el problema. En lugar de tener dos índices que siempre aumentan, ahora tendremos dos índices. **Uno siempre aumenta, el otro siempre disminuye**. Por lo tanto también haremos **a lo sumo 2n pasos**. Complejidad $\mathcal{O}(n)$.

Entrada

Buscamos sumar $x = 12$

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 1

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x$$

$$\begin{array}{cccccccc} & i & & & & & & j \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 1

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x \rightarrow \text{Aumento } i$$

$$\begin{array}{cccccccc} & i & & & & & & j \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 2

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{14} > x$$

$$A = \begin{matrix} & & i & & & & & j \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 2

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{14} > x \rightarrow \text{Disminuyo } j$$

$$A = \begin{matrix} & & i & & & & & j \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 3

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x$$

$$\begin{array}{cccccccc} & & i & & & & j & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 3

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x \rightarrow \text{Disminuyo } j$$

$$A = \begin{matrix} & & i & & & & & j & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 4

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x$$

$$\begin{array}{cccccccc} & & i & & & j & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 4

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x \rightarrow \text{Disminuyo } j$$

$$\begin{array}{cccccccc} & & i & & & j & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 5

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x$$

$$\begin{array}{cccccccc} & & i & & & j & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$
$$A = [1, 4, 5, 6, 7, 9, 9, 10]$$

Paso 5

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x \rightarrow \text{Aumento } i$$

$$A = \begin{matrix} & & i & & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 6

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{12} = x$$

$$A = \begin{matrix} & & i & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$

Paso 6

Buscamos sumar $x = 12$

$$\overbrace{A[i] + A[j]}^{12} = x \rightarrow \text{Podemos terminar}$$

$$\begin{array}{c} \\ A = \end{array} \begin{array}{cccccccc} & & i & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{array}$$

Código

Código

```
// En iR, jR devolvemos la respuesta
void ventanaDeslizante(vector<int> &A, int x, int &iR,
{                                                int &jR)
    int n = int(A.size()), j = n-1;
    for(int i = 0; i < n; i++) // fijando el extremo i:
    { // mientras la suma sea > x, disminuimos j
        while (j > i && A[i] + A[j] > x)
            j--;
        // Sale del while : (A[i] + A[j] <= x) o (j == i)
        if (j > i && A[i] + A[j] == x)
        {
            iR = i;
            jR = j;
        }
    }
}
```

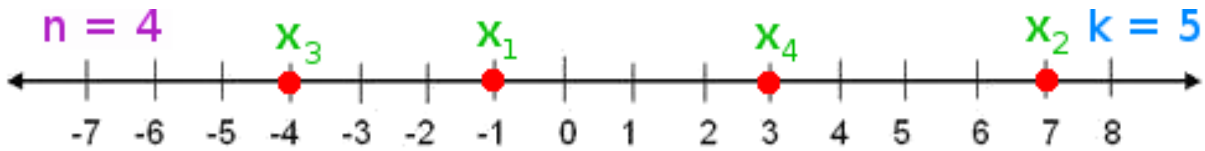
¿Preguntas?

Problema:

Dados n puntos en la recta numérica ($A \leq x_1, x_2, \ldots, x_n \leq B$), y un número k. Si listamos las distancias entre pares de puntos (x_i, x_j) con $i < j$ y ordenamos estas distancias (puede haber repetidas). Buscamos qué distancia quedó en el k-ésimo lugar.

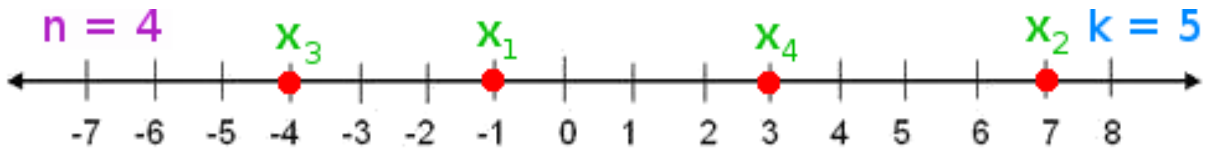
Problema:

Dados n puntos en la recta numérica ($A \leq x_1, x_2, \ldots, x_n \leq B$), y un número k. Si listamos las distancias entre pares de puntos (x_i, x_j) con $i < j$ y ordenamos estas distancias (puede haber repetidas). Buscamos qué distancia quedó en el k-ésimo lugar.



Problema:

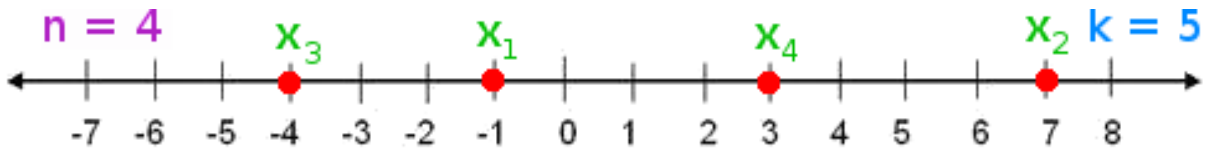
Dados n puntos en la recta numérica ($A \leq x_1, x_2, \ldots, x_n \leq B$), y un número k. Si listamos las distancias entre pares de puntos (x_i, x_j) con $i < j$ y ordenamos estas distancias (puede haber repetidas). Buscamos qué distancia quedó en el k-ésimo lugar.



1. $\text{dist}(x_1, x_2) = 8$
2. $\text{dist}(x_1, x_3) = 3$
3. $\text{dist}(x_1, x_4) = 4$
4. $\text{dist}(x_2, x_3) = 11$
5. $\text{dist}(x_2, x_4) = 4$
6. $\text{dist}(x_3, x_4) = 7$

Problema:

Dados n puntos en la recta numérica ($A \leq x_1, x_2, \ldots, x_n \leq B$), y un número k. Si listamos las distancias entre pares de puntos (x_i, x_j) con $i < j$ y ordenamos estas distancias (puede haber repetidas). Buscamos qué distancia quedó en el k-ésimo lugar.

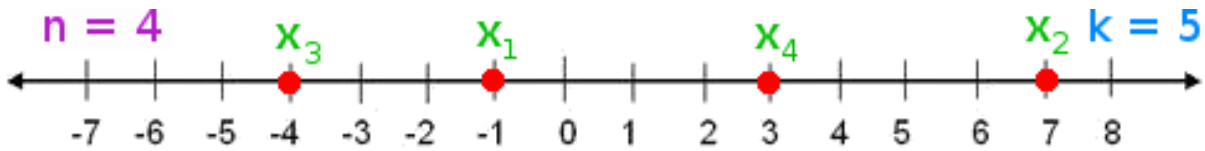


1. $\text{dist}(x_1, x_2) = 8$
2. $\text{dist}(x_1, x_3) = 3$
3. $\text{dist}(x_1, x_4) = 4$
4. $\text{dist}(x_2, x_3) = 11$
5. $\text{dist}(x_2, x_4) = 4$
6. $\text{dist}(x_3, x_4) = 7$

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ D = [3, & 4, & 4, & 7, & 8, & 11] \end{matrix}$$

Problema:

Dados n puntos en la recta numérica ($A \leq x_1, x_2, \ldots, x_n \leq B$), y un número k. Si listamos las distancias entre pares de puntos (x_i, x_j) con $i < j$ y ordenamos estas distancias (puede haber repetidas). Buscamos qué distancia quedó en el k-ésimo lugar.



1. $\text{dist}(x_1, x_2) = 8$
2. $\text{dist}(x_1, x_3) = 3$
3. $\text{dist}(x_1, x_4) = 4$
4. $\text{dist}(x_2, x_3) = 11$
5. $\text{dist}(x_2, x_4) = 4$
6. $\text{dist}(x_3, x_4) = 7$

$$D = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ [3, & 4, & 4, & 7, & 8, & 11] \end{matrix}$$

Solución

Solución

1. Calcular todas las distancias, ordenarlas, y ver qué elemento está en el k-ésimo lugar. Complejidad : $\mathcal{O}(n^2 + n^2 \lg(n))$

Solución

1. Calcular todas las distancias, ordenarlas, y ver qué elemento está en el k-ésimo lugar. Complejidad : $\mathcal{O}(n^2 + n^2 \lg(n))$
2. Usar **Binary Search** y una **Ventana deslizante**.

Solución

1. Calcular todas las distancias, ordenarlas, y ver qué elemento está en el k-ésimo lugar. Complejidad : $\mathcal{O}(n^2 + n^2 \lg(n))$
2. Usar **Binary Search** y una **Ventana deslizante**. Haremos búsqueda binaria en la distancia. Formalmente, la propiedad será :
   $P(d)$: La cantidad de distancias menores que **d** es menor que k

Solución

1. Calcular todas las distancias, ordenarlas, y ver qué elemento está en el k-ésimo lugar. Complejidad : $\mathcal{O}(n^2 + n^2 \lg(n))$
2. Usar **Binary Search** y una **Ventana deslizante**. Haremos búsqueda binaria en la distancia. Formalmente, la propiedad será :
   $P(d)$: La cantidad de distancias menores que **d** es menor que k
   Ordenados los puntos, para una **distancia** d **fija**, podemos ir moviendo una ventana deslizante que **para cada extremo izquierdo nos diga cuántos puntos están a distancia menor que** d.

Solución

1. Calcular todas las distancias, ordenarlas, y ver qué elemento está en el k-ésimo lugar. Complejidad : $\mathcal{O}(n^2 + n^2 \lg(n))$
2. Usar **Binary Search** y una **Ventana deslizante**. Haremos búsqueda binaria en la distancia. Formalmente, la propiedad será :

   $P(d)$: La cantidad de distancias menores que **d** es menor que k

   Ordenados los puntos, para una **distancia** d **fija**, podemos ir moviendo una ventana deslizante que **para cada extremo izquierdo nos diga cuántos puntos están a distancia menor que** d.

   Complejidad :

$$\mathcal{O}(\underbrace{n \lg(n)}_{\text{Ordenar}} + \underbrace{\lg(B-A)}_{\text{BB}} \cdot \underbrace{n}_{\text{VD}}) = \mathcal{O}(n(\lg(n) + \lg(B-A))$$

¿Preguntas?

¡Lista de problemas!

¡Lista de problemas!

- Codechef - Finding Square Root (Binary Search - Dificultad fácil)
- Codeforces 287B - Pipeline (Binary Search - Dificultad fácil (suma de gauss))
- Codeforces 256B - Mr. Bender and Square (Binary Search - Dificultad media)
- Codeforces 237C - Primes on Interval - Binary Search + Ventanas deslizantes - Dificultad media (requiere criba de eratóstenes)
- Codeforces 190D - Non-Secret Cypher - Ventanas deslizantes - Dificultad media/difícil
- Codeforces 180E - Cubes - Binary Search + Ventanas deslizantes - Dificultad media/difícil

¡Gracias sponsors!

DIAMOND PLUS SPONSOR TC 2025

PLATINUM SPONSOR TC 2025

GOLD SPONSOR TC 2025

GOLD SPONSOR TC 2025