Programação Competitiva

Aula 2 - Guloso

Emanuel Juliano

Universidade Federal de Minas Gerais

14 de Julho de 2020

Motivação: Fatorial

- Dado um inteiro positivo N, você deve escrever um programa para determinar o menor número k tal que $N = a_1! + a_2! + ... + a_k!$, onde cada a_i, para $1 \leq i \leq k$, é um número inteiro positivo.

Motivação: Fatorial

- Dado um inteiro positivo N, você deve escrever um programa para determinar o menor número k tal que $N = a_1! + a_2! + ... + a_k!$, onde cada a_i, para $1 \leq i \leq k$, é um número inteiro positivo.

Exemplos de Entrada	Exemplos de Saída
10	3
25	2

XX Maratona de Programação da SBC 2015

- No exemplo, o número 10 pode ser escrito como a soma de três fatoriais $10 = 3! + 2! + 2!$ e nós vamos provar que não se pode fazer melhor!

Guloso:

- Imagine que você tem um problema de otimização sobre um conjunto de elementos (maximizar um somatório, descobrir o menor inteiro que satisfaça uma propriedade...).

Guloso:

- Imagine que você tem um problema de otimização sobre um conjunto de elementos (maximizar um somatório, descobrir o menor inteiro que satisfaça uma propriedade...).
- Mas a parte mais importante, você pode fazer escolhas sobre quais elementos constituirão sua resposta.

Guloso:

- Imagine que você tem um problema de otimização sobre um conjunto de elementos (maximizar um somatório, descobrir o menor inteiro que satisfaça uma propriedade...).
- Mas a parte mais importante, você pode fazer escolhas sobre quais elementos constituirão sua resposta.
- Aquilo que o algoritmo guloso faz é realizar escolhas localmente ótimas de maneira que essas se combinem em uma solução globalmente ótima.

Usando Guloso

- Queremos representar um número **N** como a soma de **k** fatoriais, de forma a minimizar **k**.

Usando Guloso

- Queremos representar um número **N** como a soma de **k** fatoriais, de forma a minimizar **k**.

$$
\begin{aligned}
10 = & 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! \\
& 2! + 2! + 2! + 1! + 1! + 1! + 1! \\
& 3! + 1! + 1! + 1! + 1! \\
& 3! + 2! + 2!
\end{aligned}
$$

Usando Guloso

- Queremos representar um número **N** como a soma de **k** fatoriais, de forma a minimizar **k**.

$$
\begin{aligned}
10 = & 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! \\
& 2! + 2! + 2! + 1! + 1! + 1! + 1! \\
& 3! + 1! + 1! + 1! + 1! \\
& 3! + 2! + 2!
\end{aligned}
$$

- O valor **a!** não será usado mais que **a** vezes.

Usando Guloso

- Queremos representar um número **N** como a soma de **k** fatoriais, de forma a minimizar **k**.

$$
\begin{aligned}
10 = & 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! + 1! \\
& 2! + 2! + 2! + 1! + 1! + 1! + 1! \\
& 3! + 1! + 1! + 1! + 1! \\
& 3! + 2! + 2!
\end{aligned}
$$

- O valor **a!** não será usado mais que **a** vezes.
- Escolhendo gulosamente, o ótimo local é sempre escolher o **maior** fatorial.

Exercicio resolvido: Fatorial

Exercicio resolvido: Fatorial

```
1  vector<int> fat(11);
2  fat[0] = 1;
3  for(int i=1; i<=10; i++) fat[i] = fat[i-1]*i;
4
5  int n; cin >> n;
6  int ans = 0;
7  for(int i=10;i>0; i--){
8      if(n>fat[i]){
9          int at = ans/fat[i];
10         n -= at*fat[i];
11         ans += at;
12     }
13 }
14 cout << ans << endl;
```

Motivação: Restaurant

- Um restaurante recebe **N** ($N \leq 10^5$) pedidos de reserva. Cada pedido ocupa o restaurante por um período contíguo de l_i até r_i, por exemplo, das 9 às 10 horas. Com a restrição de que duas reservas não podem ter interseção, qual o maior número de pedidos que podemos aceitar?

Motivação: Restaurant

- Um restaurante recebe **N** ($N \leq 10^5$) pedidos de reserva. Cada pedido ocupa o restaurante por um período contíguo de l_i até r_i, por exemplo, das 9 às 10 horas. Com a restrição de que duas reservas não podem ter interseção, qual o maior número de pedidos que podemos aceitar?

input Copy

```
5
1 2
2 3
3 4
4 5
5 6
```

output Copy

```
3
```

Ordenação

- Existe uma série de problemas gulosos que são baseados em ordenar os elementos, dependendo de como eles forem representados:

Ordenação

- Existe uma série de problemas gulosos que são baseados em ordenar os elementos, dependendo de como eles forem representados:
- Vetor de **números**: ordenar de forma crescente/decrescente e, sempre que possível, adicionando os valores na resposta. Exemplo: Fatorial.

Ordenação

- Existe uma série de problemas gulosos que são baseados em ordenar os elementos, dependendo de como eles forem representados:
- Vetor de **números**: ordenar de forma crescente/decrescente e, sempre que possível, adicionando os valores na resposta. Exemplo: Fatorial.
- Vetor de **palavras**: ordenar lexicograficamente

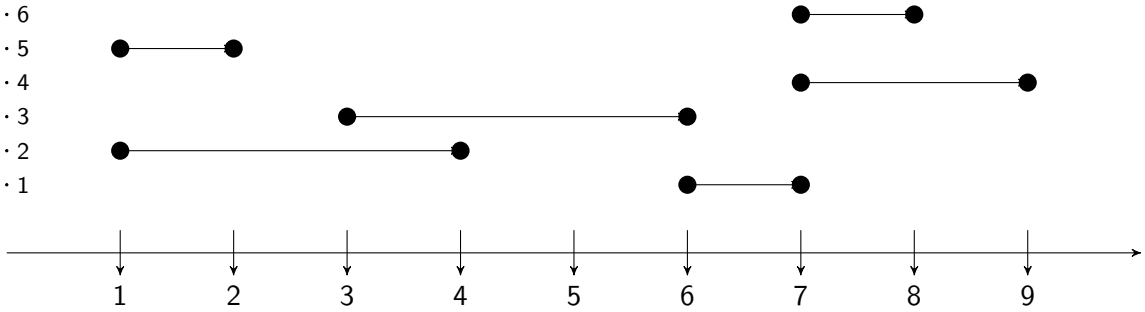
Ordenação

- Existe uma série de problemas gulosos que são baseados em ordenar os elementos, dependendo de como eles forem representados:
- Vetor de **números**: ordenar de forma crescente/decrescente e, sempre que possível, adicionando os valores na resposta. Exemplo: Fatorial.
- Vetor de **palavras**: ordenar lexicograficamente
- Vetor de **pares**: ordenar pela diferença (second-first), ordenar pelo primeiro/segundo elemento, ordenar por alguma fórmula $f(x, y)$.

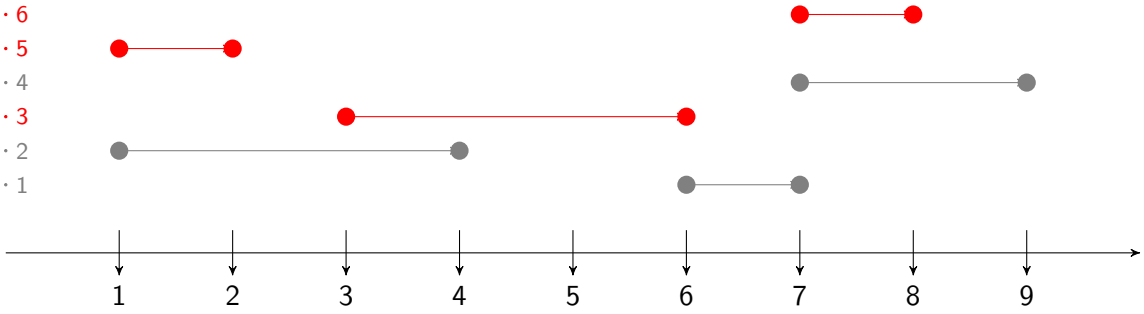
Ordenação

- Existe uma série de problemas gulosos que são baseados em ordenar os elementos, dependendo de como eles forem representados:
- Vetor de **números**: ordenar de forma crescente/decrescente e, sempre que possível, adicionando os valores na resposta. Exemplo: Fatorial.
- Vetor de **palavras**: ordenar lexicograficamente
- Vetor de **pares**: ordenar pela diferença (second-first), ordenar pelo primeiro/segundo elemento, ordenar por alguma fórmula $f(x, y)$.
- Aquilo que importa é deixar o seu vetor propício às suas escolhas gulosas funcionarem.

Sendo guloso no restaurante



Sendo guloso no restaurante



Sendo guloso no restaurante

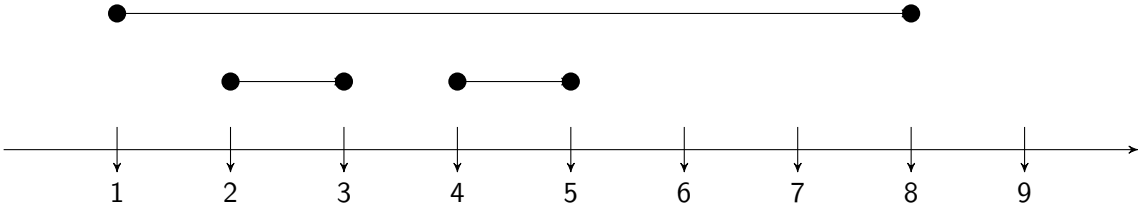
- Vamos pensar em alguma ideia de ordenação para os intervalos:

Sendo guloso no restaurante

- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo?

Sendo guloso no restaurante

- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo?

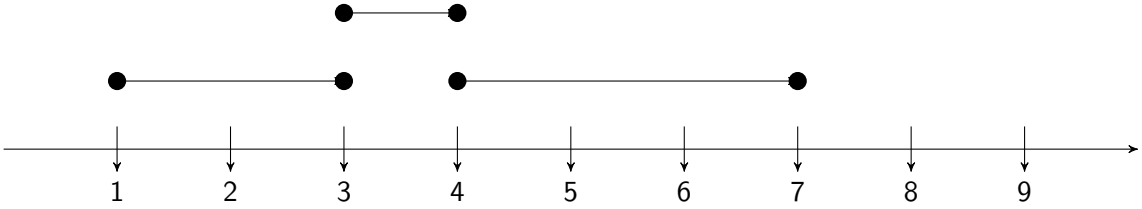


Sendo guloso no restaurante

- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo? Falha
- Ordenar pelo tamanho?

Sendo guloso no restaurante

- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo? Falha
- Ordenar pelo tamanho?

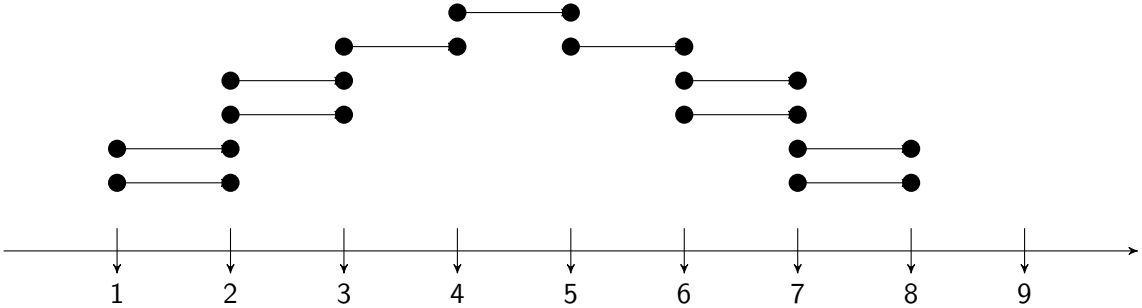


Sendo guloso no restaurante

- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo? Falha
- Ordenar pelo tamanho? Falha
- Ordenar pelo número de interseções?

Sendo guloso no restaurante

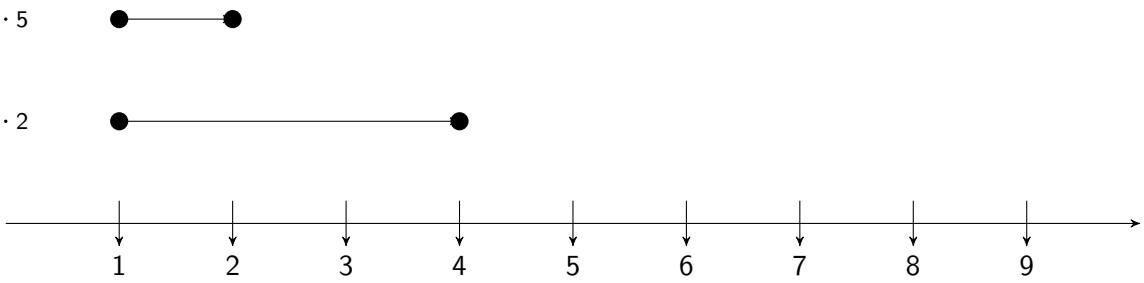
- Vamos pensar em alguma ideia de ordenação para os intervalos:
- Ordenar pelo começo? Falha
- Ordenar pelo tamanho? Falha
- Ordenar pelo número de interseções?



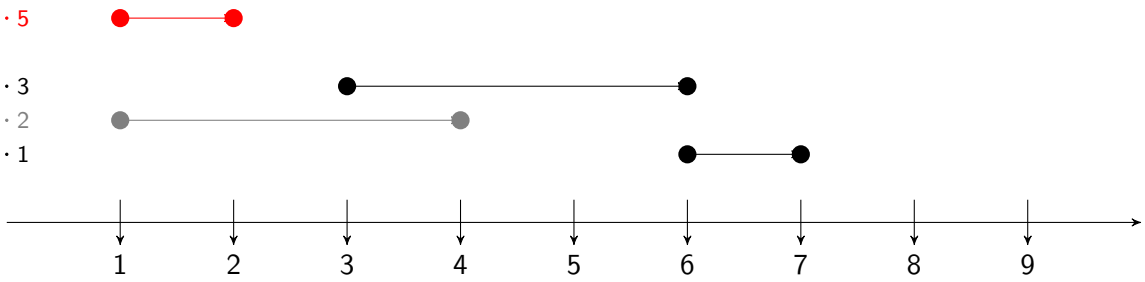
Sendo guloso no restaurante

- Vamos pensar em algumas ideias para a ordenação:
- Ordenar pelo começo do intervalo? Falha
- Ordenar pelo tamanho? Falha
- Ordenar pelo número de interseções? Falha
- Ordenar pelo final do intervalo?

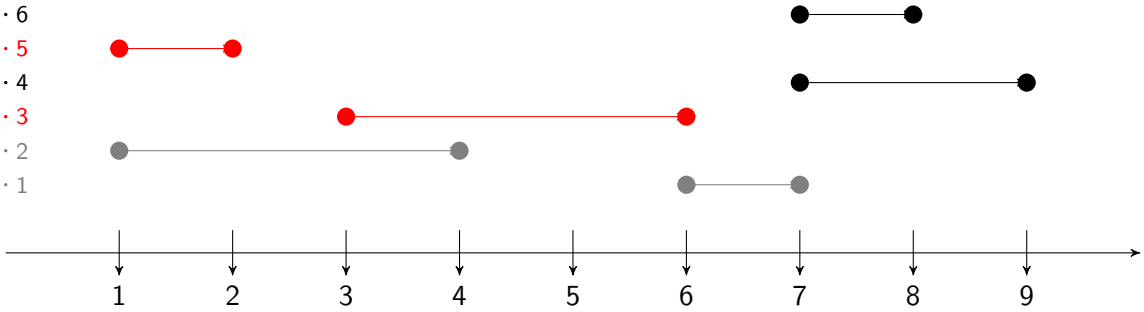
Sendo guloso no restaurante



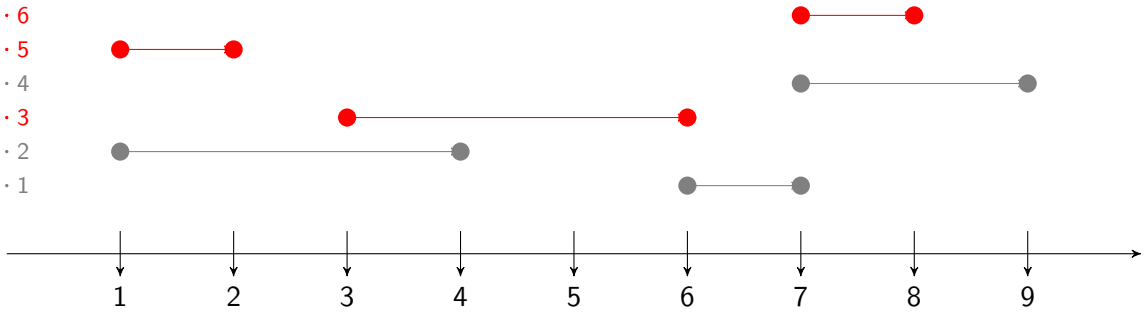
Sendo guloso no restaurante



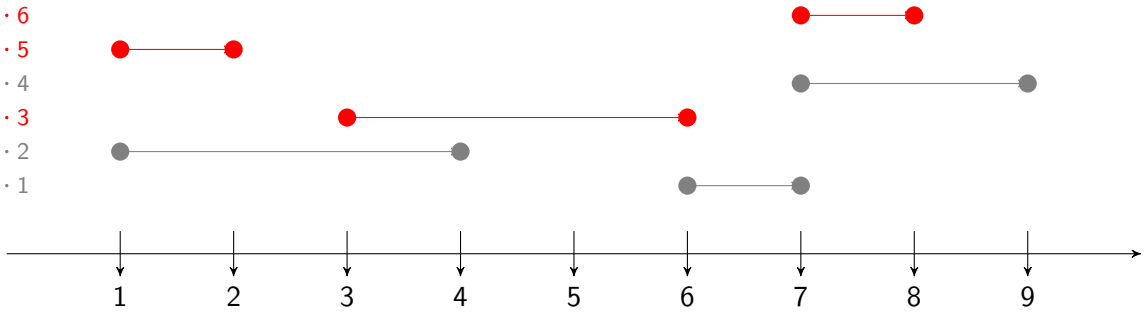
Sendo guloso no restaurante



Sendo guloso no restaurante



Sendo guloso no restaurante



- Ordenar pelo fim parece uma boa ideia, vamos tentar provar.

Sendo guloso no restaurante - Prova

- Imagine que alguém afirme que possui uma resposta melhor que a nossa.

Sendo guloso no restaurante - Prova

- Imagine que alguém afirme que possui uma resposta melhor que a nossa.
- Nesse caso, assuma que antes de uma certa posição **i**, ele escolheu os mesmos intervalos da nossa resposta, ou seja, essa é a primeira posição que ele faz uma escolha diferente.

Sendo guloso no restaurante - Prova

- Imagine que alguém afirme que possui uma resposta melhor que a nossa.
- Nesse caso, assuma que antes de uma certa posição **i**, ele escolheu os mesmos intervalos da nossa resposta, ou seja, essa é a primeira posição que ele faz uma escolha diferente.
- Sendo assim, esse intervalo termina depois do nosso. Nesse caso, substituí-lo pela nossa escolha ainda garante o mesmo resultado, pois ainda não estamos criando nenhuma interseção.

Sendo guloso no restaurante - Prova

- Imagine que alguém afirme que possui uma resposta melhor que a nossa.
- Nesse caso, assuma que antes de uma certa posição **i**, ele escolheu os mesmos intervalos da nossa resposta, ou seja, essa é a primeira posição que ele faz uma escolha diferente.
- Sendo assim, esse intervalo termina depois do nosso. Nesse caso, substituí-lo pela nossa escolha ainda garante o mesmo resultado, pois ainda não estamos criando nenhuma interseção.
- Logo, nossa resposta é no minimo tão boa e, assim, não existe solução melhor.

Exercicio resolvido: Resturant - Comparador

Exercicio resolvido: Resturant - Comparador

- Como várias vezes teremos que ordenar os elementos de um vetor seguindo diferentes critérios, precisamos como fazer isso.

Exercicio resolvido: Resturant - Comparador

- Como várias vezes teremos que ordenar os elementos de um vetor seguindo diferentes critérios, precisamos como fazer isso.
- Em C++, para ordenar o vetor como quisermos, podemos criar uma função booleana, que nos diz quando um número é "menor"que o outro.

Exercicio resolvido: Resturant - Comparador

- Como várias vezes teremos que ordenar os elementos de um vetor seguindo diferentes critérios, precisamos como fazer isso.
- Em C++, para ordenar o vetor como quisermos, podemos criar uma função booleana, que nos diz quando um número é "menor"que o outro.

```
1  bool cmp(pair<int, int> a, pair<int, int> b){
2      if(a.s != b.s) return a.s < b.s;
3      else return a.f < b.f;
4  }
5
6  // sort(v.begin(), v.end(), cmp);
```

Exercicio resolvido: Resturant - Main

Exercicio resolvido: Resturant - Main

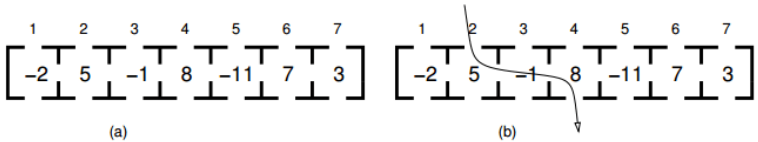
```
1  int n; cin >> n;
2  vector<pair<int, int>> v;
3  for(int i=0;i <n; i++){
4      int a, b; cin >> a >> b;
5      v.push_back({a, b});
6  }
7  sort(v.begin(), v.end(), cmp);
8
9  int last = -1, ans = 0;
10 for(auto p : v){
11     int r = p.s, l = p.f;
12     if(l>last) last = r, ans++;
13 }
14 cout << ans << endl;
```

Motivação: Corredor

- Bruninho tem que entrar em um corredor, percorrer algumas salas e depois sair do corredor. Ele pode entrar apenas uma vez, e passar por cada sala apenas uma vez. Ao passar por uma sala o ele ganha um certo número de vidas (que pode ser negativo!). O objetivo é passar pelo corredor coletando a maior quantidade possível de vidas!

Motivação: Corredor

- Bruninho tem que entrar em um corredor, percorrer algumas salas e depois sair do corredor. Ele pode entrar apenas uma vez, e passar por cada sala apenas uma vez. Ao passar por uma sala o ele ganha um certo número de vidas (que pode ser negativo!). O objetivo é passar pelo corredor coletando a maior quantidade possível de vidas!



Exemplos de Entrada	Exemplos de Saída
7 -2 5 -1 8 -11 7 3	12
10 50 42 -35 2 -60 5 30 -1 40 31	105

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 25$$

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 25$$
$$\{\ldots,\ -3,\ \ldots\} \qquad \text{soma em i} = 12$$

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 25$$
$$\{\ldots,\ -3,\ \ldots\} \qquad \text{soma em i} = 12$$

  - soma máxima para i-1 = -15

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 25$$
$$\{\ldots,\ -3,\ \ldots\} \qquad \text{soma em i} = 12$$

  - soma máxima para i-1 = -15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 10$$

Observações:

- Se o vetor for todo positivo, a resposta será a soma de todos os elementos do vetor.

$$\{2,\ 5,\ 4,\ 1,\ 6,\ 2\} \qquad \text{ans} = 20$$

- Se o vetor for todo de números negativos, a resposta será o maior elemento do vetor

$$\{-2,\ -10,\ -6,\ -10\} \qquad \text{ans} = -2$$

- Soma máxima que termina na posição i só depende da soma máxima na posição i-1 e do valor v[i]
  - soma máxima para i-1 = 15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 25$$
$$\{\ldots,\ -3,\ \ldots\} \qquad \text{soma em i} = 12$$

  - soma máxima para i-1 = -15

$$\{\ldots,\ 10,\ \ldots\} \qquad \text{soma em i} = 10$$
$$\{\ldots,\ -3,\ \ldots\} \qquad \text{soma em i} = -3$$

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{$`-2, 5, -1, 8, -13, 7, 3`$\}$ soma_ant = -INF ans = -INF

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{-2, 5, -1, 8, -13, 7, 3\}$ soma_ant = -2 ans = -2

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{$`-2, 5, -1, 8, -13, 7, 3`$\}$ soma_ant $= 5$ ans $= 5$

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{$`-2, 5, -1, 8, -13, 7, 3`$\}$ soma_ant $= 4$ ans $= 5$

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{-2, 5, -1, 8, -13, 7, 3\}$ soma_ant $= 12$ ans $= 12$

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{-2, 5, -1, 8, -13, 7, 3\}$ soma_ant = -1 ans = 12

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{$`-2, 5, -1, 8, -13, 7, 3`$\}$ soma_ant $= 7$ ans $= 12$

Kadane

- Manteremos uma variável **soma_ant** que nos diz o valor da maior soma que termina na posição anterior (i-1).
- Se soma_ant for maior que 0, a soma máxima de i será v[i]+soma_ant
- Se soma_ant for menor que 0, a soma máxima de i será v[i]
- Atualizamos soma_ant e vemos se é maior que nossa resposta

$\{$`-2, 5, -1, 8, -11, 7, 3`$\}$ soma_ant $= 10$ ans $= 12$

Exercicio resolvido: Corredor

Exercicio resolvido: Corredor

```
1  int n; cin >> n;
2  vector<int> v(n);
3  for(int i=0;i <n; i++) cin >> v[i];
4
5  int ans=-INF, soma_at=-INF;
6  for(int i=0;i <n; i++){
7      if(soma_at<0) soma_at = v[i];
8      else soma_at += v[i];
9
10     ans = max(ans, soma_at);
11 }
12 cout << ans << endl;
```

Material

- Lista de Exercícios (“Guloso”)
- Neps Academy:
  - Algoritmo Guloso
  - Kadane
- Geeks for Geeks