Exercício 1

Exercício (CLRS Problem 9-1. Dado um conjunto de n números, queremos **listar em ordem crescente** os i maiores elementos deste usando um algoritmo baseado em comparações.
Compare a complexidade dos seguintes métodos em função de n e i.

- Ordene o vetor e liste os i maiores elementos.
- Construa uma fila de prioridade (max-heap) e chame a rotina EXTRACT-MAX i vezes.
- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele e ordene os i maiores elementos.

Observação: note que i pode ser uma função de n. Por exemplo, $i = 43$, $i = \lceil \lg n \rceil$ ou $i = \lfloor n/2 \rfloor$.

Exercício 1 – Solução

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade:

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina Extract-Max i vezes.

  Complexidade:

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina Extract-Max i vezes.

  Complexidade: $O(n + i \lg n)$.

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina Extract-Max i vezes.

  Complexidade: $O(n + i \lg n)$.

- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele, ordene os i maiores elementos e liste os elementos.

  Complexidade:

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina Extract-Max i vezes.

  Complexidade: $O(n + i \lg n)$.

- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele, ordene os i maiores elementos e liste os elementos.

  Complexidade: $O(n) + O(n) + O(i \lg i) + O(i) = O(n + i \lg i)$.

Comparação dos métodos:

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina EXTRACT-MAX i vezes.

  Complexidade: $O(n + i \lg n)$.

- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele, ordene os i maiores elementos e liste os elementos.

  Complexidade: $O(n) + O(n) + O(i \lg i) + O(i) = O(n + i \lg i)$.

Comparação dos métodos:

- $i = \Theta(n)$: todos os métodos têm a mesma complexidade.

Exercício 1 – Solução

- Ordene o vetor e liste os i maiores elementos.

  Complexidade: $O(n \lg n + i)$.

- Construa uma fila de prioridade (max-heap) e chame a rotina Extract-Max i vezes.

  Complexidade: $O(n + i \lg n)$.

- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele, ordene os i maiores elementos e liste os elementos.

  Complexidade: $O(n) + O(n) + O(i \lg i) + O(i) = O(n + i \lg i)$.

Comparação dos métodos:

- $i = \Theta(n)$: todos os métodos têm a mesma complexidade.
- $i = o(n)$: o terceiro método é o melhor assintoticamente.

Exercício 2

Exercício (CLRS 9.3-5). Suponha que você tenha uma subrotina do tipo "caixa-preta" que determina a mediana em tempo linear (no pior caso). Descreva um algoritmo linear simples que resolve o problema da seleção para qualquer i dado na entrada.

Observação: suponha que o algoritmo $\textsc{Mediana}(A, p, r)$ devolve um **índice** q tal que $A[q]$ é a mediana (inferior, digamos) de $A[p\,..\,r]$ (possivelmente rearranjando os elementos de A).

Escreva o pseudo-código de um algoritmo linear $\textsc{Select}(A, p, r, i)$ que devolve o (ou o índice do) i-ésimo menor elemento de $A[p\,..\,r]$.

```
SELECT(A, p, r, i)
    se p = r
        então devolva p
    q ← MEDIANA(A, p, r)
    A[q] ↔ A[r]
    q ← PARTICIONE(A, p, r)
    k ← q − p + 1
    se i = k
        então devolva q
        senão se i < k
            então devolva SELECT(A, p, q − 1, i)
            senão devolva SELECT(A, q + 1, r, i − k)
```

Exercício 2 – Solução

```
SELECT(A, p, r, i)
se p = r
    então devolva p
q ← MEDIANA(A, p, r)
A[q] ↔ A[r]
q ← PARTICIONE(A, p, r)
k ← q − p + 1
se i = k
    então devolva q
    senão se i < k
        então devolva SELECT(A, p, q − 1, i)
        senão devolva SELECT(A, q + 1, r, i − k)
```

Recorrência:

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + \Theta(n) & n > 1 \end{cases}$$

$T(n) = O(??)$.

Exercício 2 – Solução

Suponha que $T(n) = T(n/2) + cn$.

Método da árvore de recorrência:

Nível 0: custo cn

Nível 1: custo $c(n/2)$

Nível 2: custo $c(n/4)$

Nível i: custo $c(n/2^i)$

Assim,

$$\begin{aligned} T(n) &= cn + cn/2 + cn/4 + \cdots + T(1) \\ &= (cn + cn/2 + cn/4 + \cdots) + T(1) \\ &\leq cn(1 + 1/2 + 1/4 + \ldots) + T(1) \\ &= cn\frac{1}{(1-1/2)} + T(1) = 2cn + T(1) \end{aligned}$$

Exercício 2 – Solução

Suponha que $T(n) = T(n/2) + cn$.

Método da árvore de recorrência:

Nível 0: custo cn
Nível 1: custo $c(n/2)$
Nível 2: custo $c(n/4)$
Nível i: custo $c(n/2^i)$
Assim,

$$\begin{aligned} T(n) &= cn + cn/2 + cn/4 + \cdots + T(1) \\ &= (cn + cn/2 + cn/4 + \cdots) + T(1) \\ &\leq cn(1 + 1/2 + 1/4 + \ldots) + T(1) \\ &= cn\frac{1}{(1-1/2)} + T(1) = 2cn + T(1) \end{aligned}$$

Exercício. Use o método da substituição para mostrar que $T(n) = O(n)$.

Exercício 3

Os habitantes da lendária cidade subterrânea de Zion se defendem de **ataques de enxames de robôs** através de uma combinação de kung fu, artilharia pesada e algoritmos eficientes! Recentemente eles ficaram interessados em métodos automatizados que os ajudem a repelir esses ataques.

Exercício 3

Os habitantes da lendária cidade subterrânea de Zion se defendem de **ataques de enxames de robôs** através de uma combinação de kung fu, artilharia pesada e algoritmos eficientes! Recentemente eles ficaram interessados em métodos automatizados que os ajudem a repelir esses ataques.

Eis como um ataque de robôs se parece.

Exercício 3

Os habitantes da lendária cidade subterrânea de Zion se defendem de **ataques de enxames de robôs** através de uma combinação de kung fu, artilharia pesada e algoritmos eficientes! Recentemente eles ficaram interessados em métodos automatizados que os ajudem a repelir esses ataques.

Eis como um ataque de robôs se parece.

- Um enxame de robôs chega durante um período de n segundos. No i-ésimo segundo, chegam x_i robôs. Baseado em dados de sensores remotos, a cidade sabe antecipadamente a sequência $x_1, \ldots, x_n$.

Exercício 3

Os habitantes da lendária cidade subterrânea de Zion se defendem de **ataques de enxames de robôs** através de uma combinação de kung fu, artilharia pesada e algoritmos eficientes! Recentemente eles ficaram interessados em métodos automatizados que os ajudem a repelir esses ataques.

Eis como um ataque de robôs se parece.

- Um enxame de robôs chega durante um período de n segundos. No i-ésimo segundo, chegam x_i robôs. Baseado em dados de sensores remotos, a cidade sabe antecipadamente a sequência $x_1, \ldots, x_n$.
- A cidade tem à disposição um pulso eletromagnético (PEM) que pode destruir alguns dos robôs no instante em que eles chegam.
  A potência do PEM depende de quão longo foi o tempo de recarga.

Exercício 3

- Mais precisamente, existe uma função $f(\cdot)$ tal que se j segundos se passaram desde a última vez que o PEM foi usado, então ele é capaz de destruir até $f(j)$ robôs.
- Assim, se o PEM é usado no segundo i-ésimo e se passaram j segundos desde a última vez que foi usado, então ele destruirá $\min\{x_i, f(j)\}$ robôs. Após o uso do PEM, começa um novo período de recarga.
- Suponha que o PEM começa inicialmente descarregado. Assim, se ele for usado pela primeira vez no segundo j, ele destruirá $\min\{x_j, f(j)\}$ robôs.

Exercício 3

Exemplo: Considere a seguinte instância de tamanho $n = 4$:

i	1	2	3	4
x_i	1	10	10	1
$f(i)$	1	2	4	8

Uma solução ótima é ativar o PEM nos segundos 3 e 4. No segundo 3, o PEM destruirá $\min\{x_3, f(3)\} = \min\{10, 4\} = 4$ robôs. No segundo 4, como se passou um segundo desde a última ativação, o PEM destruirá $\min\{x_4, f(1)\} = \min\{1, 1\} = 1$. O total de robôs destruídos é $4 + 1 = 5$.

Exercício 3

Resumo da ópera:

- No segundo i chegam x_i robôs, $i = 1, 2, \ldots, n$.
- Se o PEM é usado no segundo i e se passaram j segundos desde a última vez que foi usado, então ele destruirá $\min\{x_i, f(j)\}$ robôs;
- O PEM começa inicialmente descarregado. Se ele for usado pela primeira vez no j-ésimo segundo, ele destruirá $\min\{x_j, f(j)\}$ robôs.

Exercício 3

Resumo da ópera:

- No segundo i chegam x_i robôs, $i = 1, 2, \ldots, n$.
- Se o PEM é usado no segundo i e se passaram j segundos desde a última vez que foi usado, então ele destruirá $\min\{x_i, f(j)\}$ robôs;
- O PEM começa inicialmente descarregado. Se ele for usado pela primeira vez no j-ésimo segundo, ele destruirá $\min\{x_j, f(j)\}$ robôs.

Problema: Projete um algoritmo de **programação dinâmica** que dadas a sequência $x_1, \ldots, x_n$ e a função $f(\cdot)$, determina em tempo $O(n^2)$ os instantes (segundos) em que o PEM deve ser usado de modo a destruir o maior número possível de robôs.

Exercício 3

Resumo da ópera:

- No segundo i chegam x_i robôs, $i = 1, 2, \ldots, n$.
- Se o PEM é usado no segundo i e se passaram j segundos desde a última vez que foi usado, então ele destruirá $\min\{x_i, f(j)\}$ robôs;
- O PEM começa inicialmente descarregado. Se ele for usado pela primeira vez no j-ésimo segundo, ele destruirá $\min\{x_j, f(j)\}$ robôs.

Problema: Projete um algoritmo de **programação dinâmica** que dadas a sequência $x_1, \ldots, x_n$ e a função $f(\cdot)$, determina em tempo $O(n^2)$ os instantes (segundos) em que o PEM deve ser usado de modo a destruir o maior número possível de robôs.

Sugestão. Considere uma sequência $x_1, x_2, \ldots, x_i$. Seja $r[i]$ o valor da solução ótima para esta sequência. Note que sempre vale à pena usar o PEM no instante i. Se a última vez que o PEM foi usado foi no segundo $i - j$, então no segundo i o PEM destrói $\min\{x_i, f(j)\}$ robôs. Assim $r[i] = \max_{???}(r[???] + ???)$.

Exercício 3 – Solução

Subestrutura ótima:

Exercício 3 – Solução

Subestrutura ótima:

- Suponha que conhecemos uma solução ótima $s_1, \ldots, s_k$ do problema, i.e., os segundos em que o PEM deve ser usado para destruir o maior número possível de robôs. Note que $s_k = n$, i.e., o PEM é usado no segundo n.

Exercício 3 – Solução

Subestrutura ótima:

- Suponha que conhecemos uma solução ótima $s_1, \ldots, s_k$ do problema, i.e., os segundos em que o PEM deve ser usado para destruir o maior número possível de robôs. Note que $s_k = n$, i.e., o PEM é usado no segundo n.
- O PEM destrói $\min\{x_n, f(j)\}$ robôs onde j é o número de segundos que se passaram desde o último uso do PEM. Ou seja, o PEM foi usado anteriormente no segundo $s_{k-1} = n - j$.

Exercício 3 – Solução

Subestrutura ótima:

- Suponha que conhecemos uma solução ótima $s_1, \ldots, s_k$ do problema, i.e., os segundos em que o PEM deve ser usado para destruir o maior número possível de robôs. Note que $s_k = n$, i.e., o PEM é usado no segundo n.
- O PEM destrói $\min\{x_n, f(j)\}$ robôs onde j é o número de segundos que se passaram desde o último uso do PEM. Ou seja, o PEM foi usado anteriormente no segundo $s_{k-1} = n - j$.
- Então $s_1, \ldots, s_{k-1}$ é uma solução ótima para a sequência $x_1, \ldots, x_{n-j}$. (Certo?)

1	2	3	4	5	6	7	8	9	...	$n-j$	...	...	...	n
	●			●				●	...	●				●

Exercício 3 – Solução

Troque n por i para obtermos a recorrência geral.
Seja $r[i]$ o valor ótimo do subproblema $x_1, \ldots, x_i$, i.e., o número máximo de robôs que são destruídos.

Exercício 3 – Solução

Troque n por i para obtermos a recorrência geral.
Seja $r[i]$ o valor ótimo do subproblema $x_1, \ldots, x_i$, i.e., o número máximo de robôs que são destruídos.

Podemos definir $r[0] = 0$.

Exercício 3 – Solução

Troque n por i para obtermos a recorrência geral.
Seja $r[i]$ o valor ótimo do subproblema $x_1, \ldots, x_i$, i.e., o número máximo de robôs que são destruídos.

Podemos definir $r[0] = 0$.

Se no segundo i se passaram j segundos desde o último uso do PEM, então $r[i] = r[i-j] + \min\{x_i, f(j)\}$.

1	2	3	4	5	6	7	8	9	$\ldots$	$i-j$	$\ldots$	$\ldots$	$\ldots$	i
	•			•				•	$\ldots$	•				•

Exercício 3 – Solução

Se no segundo i se passaram j segundos desde o último uso, então $r[i] = r[i-j] + \min\{x_i, f(j)\}$.

Exercício 3 – Solução

Se no segundo i se passaram j segundos desde o último uso, então $r[i] = r[i-j] + \min\{x_i, f(j)\}$.

Como não se sabe a priori o valor de j, segue que:

$$r[i] = \begin{cases} 0 & \text{i=0}, \\ \max_{j=1,\ldots,i} \left(r[i-j] + \min\{x_i, f(j)\} \right) & i > 0. \end{cases}$$

1	2	3	4	5	6	7	8	9	...	$i-j$	...	...	...	i
	●			●				●	...	●				●

Exercício 3 – Solução

Se no segundo i se passaram j segundos desde o último uso, então $r[i] = r[i-j] + \min\{x_i, f(j)\}$.

Como não se sabe a priori o valor de j, segue que:

$$r[i] = \begin{cases} 0 & \text{i=0}, \\ \max_{j=1,\ldots,i} \left(r[i-j] + \min\{x_i, f(j)\} \right) & i > 0. \end{cases}$$

1	2	3	4	5	6	7	8	9	$\ldots$	$i-j$	$\ldots$	$\ldots$	$\ldots$	i
	•			•				•	$\ldots$	•				•

Note que escolher $j = i$ significa que o PEM é usado apenas uma vez no segundo i.

Exercício 3 – Solução

$\text{PEM}(x, f, n)$

$r[0] \leftarrow 0$
para $i \leftarrow 1$ **até** n **faça**
    $q \leftarrow -\infty$
    para $j \leftarrow 1$ **até** i **faça**
        se $q < r[i-j] + \min\{x_i, f(j)\}$
            então $q \leftarrow r[i-j] + \min\{x_i, f(j)\}, \quad s[i] \leftarrow i-j$
    $r[i] \leftarrow q$
devolva r, s

Exercício 3 – Solução

$\mathrm{PEM}(x, f, n)$

$r[0] \leftarrow 0$
para $i \leftarrow 1$ **até** n **faça**
    $q \leftarrow -\infty$
    para $j \leftarrow 1$ **até** i **faça**
        se $q < r[i-j] + \min\{x_i, f(j)\}$
            então $q \leftarrow r[i-j] + \min\{x_i, f(j)\}, \quad s[i] \leftarrow i - j$
    $r[i] \leftarrow q$
devolva r, s

Complexidade: $O(n^2)$.

Exercício 3 – Solução

```
PEM(x, f, n)
r[0] ← 0
para i ← 1 até n faça
    q ← −∞
    para j ← 1 até i faça
        se q < r[i − j] + min{x_i, f(j)}
            então q ← r[i − j] + min{x_i, f(j)},  s[i] ← i − j
    r[i] ← q
devolva r, s
```

Complexidade: $O(n^2)$.

$s[i]$ denota o instante do uso anterior do PEM na solução ótima de $x_1, \ldots, x_i$.

Exercício 3 – Solução

$\textsc{Imprime-Solução}(s, n)$

1. **se** $n > 0$ **então**
2. $\quad\textsc{Imprime-Solução}(s, s[n])$
3. $\quad$ Imprime n

Complexidade: $O(n)$.

Exercício 3 – Solução

$\textsc{Imprime-Solução}(s, n)$

1. **se** $n > 0$ **então**
2. $\quad\textsc{Imprime-Solução}(s, s[n])$
3. $\quad$Imprime n

Complexidade: $O(n)$.

Complexidade total: $O(n^2) + O(n) = O(n^2)$.