Sabrina Araújo da Silva nUSP: 12566182

MAC0338 - ANÁLISE DE ALGORITMOS

LISTA 6

exercícios 6 e 10

6. Descreva um algoritmo eficiente que, dado um conjunto $\{x_1, x_2, \ldots, x_n\}$ de pontos na reta real, determine o menor conjunto de intervalos fechados de comprimento um que contém todos os pontos dados. Justifique informalmente o seu algoritmo e analise a sua complexidade.

seja A uma coleção de intervalos ótimos do conjunto $X = \{x_1, x_2, \ldots, x_n\}$

```
INTERVALO-OTIMO (X, n
0  ordene X de forma que X[1] ≤ X[2] ≤ ... ≤ X[n]
1  A ← ∅
2  enquanto X ≠ ∅ faça
3      escolha por um critério guloso um intervalo i de X
4      A ← A ∪ {i}
5      X ← X \ { j ∈ X : j intersecta i }
6  devolva A
```

Agora, é preciso encontrar o critério guloso para preencher o algoritmo.

possível critério guloso:

(1) com base no primeiro ponto do conjunto determinar o intervalo que ele irá fazer parte. Por exemplo: X[1] = 0,8, então todos os X[i] seguintes tal que 0,8 ≤ X[i] ≤ 1,8 estarão no intervalo. Assim, esses pontos do conjunto serão retirados e o algoritmo irá repetir para os pontos restantes.

Portanto, temos:

```
INTERVALO-OTIMO (X, n)
0   ordene X de forma que X[1] ≤ X[2] ≤ ... ≤ X[n]
1   A ← ∅
2   i ← 1
3   inicio ← 0
4   fim ← 0
5   enquanto X ≠ ∅ faça
6       inicio ← X[i]
7       fim ← X[i] + 1
8       enquanto i ≤ n e X[i] ≤ fim
9           intervalo ← intervalo ∪ {X[i]}
10          i ← i + 1
11      A ← A ∪ intervalo
12      X ← X \ {j ∈ X : j intersecta intervalo}
13      intervalo ← ∅
14  devolva A
```

ALGORITMO

O algoritmo ordena todos os pontos do menor para o maior. Com base no primeiro ponto dessa ordem, o início e o fim do intervalo são definidos e todos os pontos do conjunto tal que início ≤ X[i] ≤ fim são encaixados nesse intervalo. Em seguida, o intervalo é colocado na coleção A de intervalos e cada item que foi encaixado nele é retirado de X. O intervalo é resetado e o algoritmo repete para as instâncias menores.

O algoritmo funciona, porque todos os pontos têm que estar necessariamente em um intervalo e ao pegar o início dos intervalos como o valor do primeiro ponto disponível garante que no intervalo de X[i] estará todos os pontos possíveis para aproveitar o intervalo.

COMPLEXIDADE: a linha 0 consome tempo $O(n \lg n)$ para ordenar; linhas 1 a 4 : $O(1)$; linhas 5 a 13 : $O(n)$. Portanto, o algoritmo consome tempo $O(n \lg n)$

10. Um pequeno negócio, digamos, uma lojinha de xerox com uma única máquina de xerox, enfrenta o seguinte problema de escalonamento. Toda manhã, eles recebem uma coleção de tarefas de seus clientes. O dono do negócio quer executar essas tarefas em sua máquina, numa ordem que mantenha os seus clientes tão satisfeitos quanto possível. A tarefa do cliente i leva t_i unidades de tempo para ser completada. Dado um escalonamento (ou seja, uma ordem das tarefas), seja C_i o momento em que a tarefa do cliente i terminou de ser executada. Suponha ainda que cada cliente i tenha uma importância para o negócio, dada pelo número w_i. A satisfação do cliente i é dependente do tempo C_i em que sua tarefa é completada. O dono do negócio deseja determinar um escalonamento que minimize a soma ponderada $\sum_{i=1}^{n} w_i C_i$.

Projete um algoritmo eficiente para resolver esse problema. Os dados são a duração $t_1, \ldots, t_n$ das n tarefas e a importância $w_1, \ldots, w_n$ de n clientes. Seu algoritmo deve produzir uma ordem das n tarefas que minimize a soma ponderada $\sum_{i=1}^{n} w_i C_i$. Mostre que seu algoritmo produz uma resposta correta.

Exemplo: Considere $n = 2$, $t_1 = 1$ e $w_1 = 10$, $t_2 = 3$ e $w_2 = 2$. Se a primeira tarefa for executada primeiro, o valor da solução é $10 \cdot 1 + 2 \cdot 4 = 18$, enquanto que se a segunda tarefa for executada primeiro, é $10 \cdot 4 + 2 \cdot 3 = 46$.

Seja A um vetor que armazena a ordem que as tarefas devem ser executadas, T o vetor da duração das tarefas e W o vetor da importância das tarefas.

Possíveis estratégias gulosas:

(1) ordenar as tarefas tal que $W[1] \geq W[2] \geq \cdots \geq W[n]$ e executar primeiro as tarefas de maior importância.

i_1 ☐ $t_1 = 5$, $w = 10$

i_2 ☐ $t_2 = 1$, $w = 3$

executando i_1 e depois i_2 $\sum_{i=1}^{n} w_i C_i = 5 \cdot 10 + 6 \cdot 3 = 68$

executando i_2 e depois i_1 $\sum_{i=1}^{n} w_i C_i = 1 \cdot 3 + 6 \cdot 10 = 63$

=> não funciona

(2) ordenar as tarefas tal que $T[1] \leq T[2] \leq \cdots \leq T[n]$ e executar as tarefas de menor duração primeiro.

i_1 ☐ $t_1 = 5$, $w = 20$

i_2 ☐ $t_2 = 1$, $w = 1$

executando i_1 e depois i_2 $\sum_{i=1}^{n} w_i C_i = 5 \cdot 20 + 6 \cdot 1 = 106$

executando i_2 e depois i_1 $\sum_{i=1}^{n} w_i C_i = 1 \cdot 1 + 6 \cdot 20 = 121$

=> não funciona

(3) ordernar as tarefas tal que $w[1]/T[1] \geq w[2]/T[2] \geq \dots \geq w[n]/T[n]$ e executar primeiro as tarefas com maior importância por tempo primeiro.

i_1 ▭ $t_1 = 5$, $w = 20$, $w/t = 4$

i_2 ▭ $t_2 = 1$, $w = 1$, $w/t = 1$

executando i_1 e depois i_2 $\sum_{i=1}^{n} w_i C_i = 5 \cdot 20 + 6 \cdot 1 = 106$

executando i_2 e depois i_1 $\sum_{i=1}^{n} w_i C_i = 1 \cdot 1 + 6 \cdot 20 = 121$

i_1 ▭ $t_1 = 5$, $w = 10$, $w/t = 2$

i_2 ▭ $t_2 = 1$, $w = 3$, $w/t = 3$

executando i_1 e depois i_2 $\sum_{i=1}^{n} w_i C_i = 5 \cdot 10 + 6 \cdot 3 = 68$

executando i_2 e depois i_1 $\sum_{i=1}^{n} w_i C_i = 1 \cdot 3 + 6 \cdot 10 = 63$

⇒ essa estratégia gulosa parece funcionar

ALGORITMO

ORDEM-ÓTIMA (W, T, A)

1 ordena W e T de modo que $w[1]/T[1] \geq w[2]/T[2] \geq \dots \geq w[n]/T[n]$

2 para $i \leftarrow 1$ até n faça

3 $\quad A[i] \leftarrow$ tarefa armazenada em $W[i]$

4 devolva A

O algoritmo funciona com a estratégia gulosa escolhida, porque dá preferência às tarefas de maior importância por tempo, assim, a lojinha de xerox estará executando sempre a tarefa que tem mais satisfação por tempo de trabalho.

- consumo de tempo da ordenação na linha 1 é $\Theta(n \lg n)$
- consumo de tempo nas linhas 2 a 4 é $\Theta(n)$
- consumo de tempo total é $\Theta(n \lg n)$