Projeto de algoritmos: uma visão prática

Aula 04

Rafael Melo
Instituto de Computação
Universidade Federal da Bahia

Algoritmos gulosos

Motivação

- Algoritmos para problemas de otimização tipicamente executam uma sequência de passos, com um conjunto de escolhas a cada passo.
- Um Algoritmo Guloso sempre escolhe a melhor opção para o momento.
- Ele faz uma escolha ótima local esperando que ela o leve a uma solução ótima global.

Elementos de algoritmos gulosos

De maneira mais geral, segue-se a sequência de passos:

1. converta o problema de otimização tal que ao se fazer uma escolha, resta apenas um subproblema a resolver;
2. prove que sempre existe uma solução ótima para o problema original que faz a escolha gulosa, então ela sempre será segura;
3. demonstre que, fazendo-se a escolha gulosa, o que resta é um subproblema cuja solução pode ser combinada à escolha gulosa de maneira a se chegar a uma solução ótima global.

Problema de seleção de atividades

Problema de seleção de atividades

- Dado o conjunto $S = \{a_1 , a_2 , . . . , a_n\}$ com n atividades propostas que desejam utilizar um dado recurso que pode ser utilizada por apenas uma atividade por vez.
- Cada atividade a_i possui:
  - instante de início s_i ;
  - instante de término f_i ;
  - satisfaz condições: $0 \leq s_i < f_i < \infty$

Problema de seleção de atividades

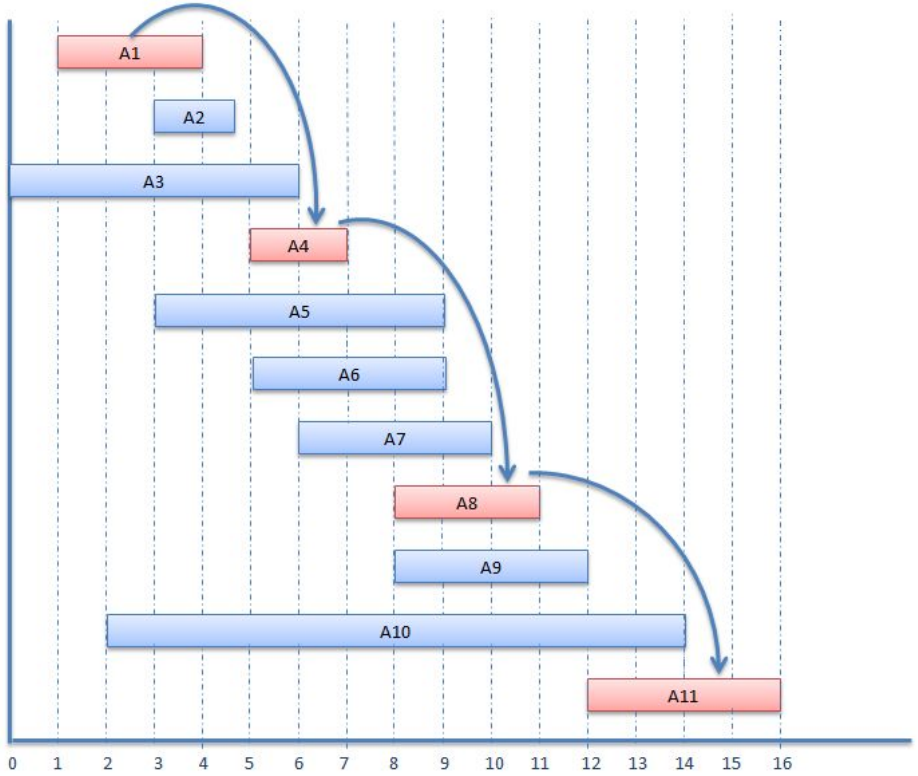
- Caso selecionada, a atividade a_i posiciona-se no intervalo semi-aberto $[s_i, f_i)$.
- Atividades a_i e a_j são compatíveis se os intervalos $[s_i, f_i)$ e $[s_j, f_j)$ não se sobrepõem.
- O **objetivo** do problema consiste em selecionar um conjunto de **tamanho máximo** de atividades **mutuamente compatíveis**.

Exemplo

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

- Alguns conjuntos de atividades mutuamente compatíveis:
  - $\{a_3, a_9, a_{11}\}$;
  - $\{a_1, a_4, a_8, a_{11}\}$, que é máximo;
  - $\{a_2, a_4, a_9, a_{11}\}$, que também é máximo.

Exemplo



Algoritmo

```python
def guloso_selecao_atividades(intervalos):
    n = len(intervalos)
    intervalos.sort(key=lambda x: x[1])

    atividades_selecionadas = [0]
    fim_ultimo_selecionado = intervalos[0][1]
    for k in range(1, n):
        inicio = intervalos[k][0]
        fim = intervalos[k][1]
        if inicio >= fim_ultimo_selecionado:
            atividades_selecionadas.append(k)
            fim_ultimo_selecionado = fim

    return atividades_selecionadas
```

Tempo de execução

- Ordenação das atividades pode ser realizada em $\Theta(n \log n)$
- Custo da seleção das atividades de forma gulosa, uma vez que as atividades já estão ordenadas por tempo de fim $\Theta(n)$

Corretude do algoritmo guloso

- Seja $S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$ o conjunto das atividades que podem iniciar após a atividade a_i terminar, e antes de a_j iniciar.
- S_{ij} consiste de todas as atividades que são compatíveis com:
  - a_i e a_j ;
  - todas as atividades que terminam antes ou no momento de término de a_i;
  - todas as atividades que iniciam no momento ou após o início de a_j.

Corretude do algoritmo guloso

- Para representar o problema completo, adicionamos atividades fictícias a_0 e a_{n+1} e adotamos a convenção $f_0 = 0$ e $s_{n+1} = \infty$.
- Então, $S = S_{0,n+1}$, e os limites para i e j são dados por $0 \leq i,j \leq n + 1$.
- Assumiremos também que as atividades estão ordenadas em ordem monotônica crescente do instante de término

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n \leq f_{n+1}$$

Corretude do algoritmo guloso

- **Proposição:** $S_{ij} = \varnothing$ se $i \geq j$
- **Demonstração:**
  - Suponha que $\exists\, a_k \in S_{ij}$, então

$$f_i \leq s_k < f_k \leq s_j < f_j \Rightarrow f_i < f_j ,$$

    o que é uma contradição.

Corretude do algoritmo guloso

Teorema: Considere qualquer subconjunto S_{ij} não vazio, e seja a_m a atividade em S_{ij} com menor instante de término:

$$f_m = \min\{f_k : a_k \in S_{ij}\}$$

Então:

1) A atividade a_m é usada em algum conjunto tamanho-máximo de atividades mutuamente compatíveis em S_{ij} .

2) O subconjunto S_{im} é vazio, e a escolha de am faz com que S_{mj} seja o único subproblema não vazio.

Corretude do algoritmo guloso

Demonstração:

- Primeiro, provaremos (2) antes porque é mais simples.
- Suponha S_{im} não vazio, de maneira que exista uma atividade a_k tal que

  $f_i \leq a_k < f_k \leq s_m < f_m$

- Então a_k está também em S_{ij} e possui um instante de término anterior a a_m , o que contradiz com a escolha.
- Concluímos então que S_{im} é vazio.

Corretude do algoritmo guloso

- Para provar a primeira parte, suponha que A_{ij} é um subconjunto tamanho-máximo de atividades mutuamente compatíveis com S_{ij}
- Ordene as atividades em A_{ij} em ordem monotonicamente crescente de instante de término.
- Seja a_k a primeira atividade em A_{ij}.
- Se $a_k = a_m$, está concluído.

Corretude do algoritmo guloso

- Se $a_k \neq a_m$, construímos o subconjunto

$$\hat{A}_{ij} = (A_{ij} - \{a_k\}) \cup \{a_m\}$$

- As atividades $\hat{A}_{ij}$ são disjuntas, pois as atividades em A_{ij} são disjuntas, a_k é a primeira atividade em A_{ij} a terminar, e $f_m \leq f_k$.
- Note que $\hat{A}_{ij}$ possui o mesmo número de atividades que A_{ij} , logo $\hat{A}_{ij}$ é um subconjunto tamanho-máximo de atividades mutuamente compatíveis para S_{ij} que inclui a_m .

Atividades práticas

Contato:
rafael.melo@ufba.br
rafaelmelo.ufba@gmail.com