Algoritmos Gulosos

prof. Fábio Luiz Usberti

MC521 - Desafios de Programação I

Instituto de Computação - UNICAMP - 2018

Sumário

Algoritmos Gulosos

Melhores escolhas locais

- Um algoritmo guloso é aquele onde em cada iteração é tomada a melhor escolha local visando atingir a solução ótima global.
- Para que um problema possa ser tratado de forma gulosa é necessário que ele apresente duas propriedades:

1. Ele deve ter sub-estrutura ótima, ou seja, a solução ótima do problema original é composta por soluções ótimas de subproblemas.
2. Ele deve apresentar a propriedade gulosa, ou seja, uma garantia teórica de que não será preciso reconsiderar uma decisão localmente ótima. Obs: a propriedade gulosa pode ser difícil de demonstrar em um ambiente de competição onde o tempo é um fator crítico.

Melhores escolhas locais

- Em geral, por ser leve, um algoritmo guloso normalmente não recebe o veredito TLE. No entanto, é comum esses algoritmos receberem o veredito WA, especialmente se não for verificada a corretude do algoritmo (propriedade gulosa).
- Provar a corretude de um algoritmo pode consumir um tempo precioso, portanto abaixo são fornecidas duas dicas na escolha da estratégia a ser adotada:

1. Se o tamanho da entrada for suficientemente pequeno para acomodar busca completa ou programação dinâmica, então é melhor garantir um veredito AC com um desses algoritmos.
2. Se o tamanho da instância for muito grande para algoritmos de busca completa e programação dinâmica conhecidos, então o algoritmo guloso tem chances de ser uma solução viável.

Dragões de Loowater

- Problema: Considere n dragões e m cavaleiros ($1 \leqslant n \leqslant m \leqslant 20000$). Um dragão i possui uma cabeça de diâmetro D_i enquanto um cavaleiro j possui uma altura H_j. A cabeça de um dragão i pode ser cortada por um cavaleiro j se $D_i \leqslant H_j$. Cada cavaleiro corta no máximo uma cabeça de dragão. Fornecida a lista de dragões e cavaleiros, é possível que todas as cabeças de dragões sejam cortadas?
- Estratégia: Esse problema pode ser resolvido de modo guloso primeiramente ordenando os tamanhos de cabeças de dragão e alturas dos cavaleiros. A escolha gulosa consiste em escolher para a menor cabeça de dragão não cortada o cavaleiro de menor altura apto para a tarefa.

Dragões de Loowater

- A complexidade dessa estratégia é dominada pelas ordenações $O(n \log n + m \log m)$.
- A sub-estrutura ótima do problema é verificada pelo fato de que em uma iteração i do algoritmo um dragão é morto, restando um subproblema contendo $n - i$ dragões. A solução ótima desse subproblema, se existir, irá compor a solução do problema original.
- A propriedade gulosa do algoritmo pode ser demonstrada pelo fato de que sempre há uma solução ótima onde a menor cabeça de dragão é cortada pelo menor cavaleiro apto para essa tarefa.

Problema do troco em moedas ("Coin Change")

- Problema: Dado um valor V e uma lista de n denominações de moedas (`coinValue[i]`), qual o número mínimo de moedas que devemos utilizar para somar o valor V. Pode-se assumir que há quantidades ilimitadas de cada denominação de moeda.
- Exemplo: $n = 4$, `coinValue = {25, 10, 5, 1}` , $V = 42$.
- Utilizando uma estratégia gulosa, descontamos do valor original a maior denominação possível que não torne o valor residual negativo, ou seja:

1. $42 - 25 = 17$
2. $17 - 10 = 7$
3. $7 - 5 = 2$
4. $2 - 1 = 1$
5. $1 - 1 = 0$

Problema do troco em moedas ("Coin Change")

- Pela estratégia gulosa utilizaremos 5 moedas, o que corresponde a solução ótima;
- Para verificar a sub-estrutura ótima desse problema, basta notar que:
  - As moedas que somam um valor 17 (subproblema do valor 42) correspondem a $10 + 5 + 1 + 1$.
  - As moedas que somam um valor 7 (subproblema do valor 17) correspondem a $5 + 1 + 1$.
  - As moedas que somam um valor 2 (subproblema do valor 7) correspondem a $1 + 1$.
  - As moedas que somam um valor 1 (subproblema do valor 2) correspondem a 1.

Problema do troco em moedas (“Coin Change”)

- Para demonstrar que o algoritmo guloso funciona, falta verificar a propriedade gulosa. Isso envolve demonstrar que, para qualquer valor V que se deseja somar, a moeda de maior denominação que seja menor do que V está em uma solução ótima.
- Essa propriedade gulosa é válida somente para alguns conjuntos de denominações de moedas. Um conjunto de denominações de moedas é denominado canônico quando apresenta propriedade gulosa.

Problema de balanceamento ("Load Balancing")

- Problema (UVa 410): Dado um conjunto de C câmaras que devem armazenar uma quantidade $1 \leqslant S \leqslant 2C$ de espécimes, tal que cada espécime j possui um peso M_j e cada câmara pode armazenar no máximo 2 espécimes. Determinar em quais câmaras os espécimes devem ser armazenados de modo a minimizar o desbalanceamento da distribuição de massas, dada pela seguinte função:

$$\min \sum_{i=1}^{C} |X_i - A|$$

- $A = \dfrac{1}{C} \sum_{j=1}^{S} M_j$ – corresponde ao peso alvo para as câmaras.
- X_i – peso total dos animais na câmara i.
- M_j – peso de um espécime j.

Problema de balanceamento ("Load Balancing")

- A solução desse problema consiste em distribuir os espécimes aos pares da forma mais equilibrada possível.
- Como nem sempre vai haver um número $2C$ de espécimes, pode-se gerar $2C - S$ espécimes virtuais de peso 0.
- Por exemplo, seja $C = 3$, $S = 4$ e $M = \{5, 1, 2, 7\}$. Nesse caso criam-se dois novos espécimes de peso 0: $M = \{5, 1, 2, 7, 0, 0\}$.
- Em seguida, os espécimes são ordenados por massa, tal que, $M_1 \leqslant M_2 \leqslant \ldots \leqslant M_{2C}$: $M = \{0, 0, 1, 2, 5, 7\}$.

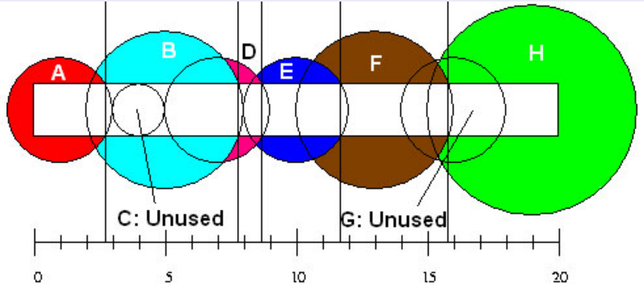
Problema de balanceamento (“Load Balancing”)

- Após a ordenação, uma estratégia gulosa torna-se aparente: adicionar o par de espécimes de menor e maior peso em uma câmera e resolver o subproblema resultante.
- Esse problema possui sub-estrutura ótima pois uma vez definido os espécimes de uma câmara, as câmaras faltantes deverão ser balanceadas no subproblema resultante e a solução ótima desse subproblema fará parte do problema original.
- Para demonstrar a propriedade gulosa é possível verificar que, escolhendo dois pares de espécimes (M_1, M_i) e (M_j, M_{2C}) obtém-se uma solução pior ou igual a escolher o par guloso (M_1, M_{2C}) e o par complementar (M_i, M_j).

Regando a grama ("Watering Grass")

- Problema (UVa 10382): Uma quantidade $n \leqslant 10000$ de aspersores são instalados em uma faixa retangular de grama de L metros de comprimento e W metros de largura. Os aspersores são posicionados colinearmente no sentido do comprimento do retângulo, na metade de sua largura. A posição de cada aspersor é fornecida a partir de sua distância da extremidade esquerda do retângulo. Além disso, também é fornecido o raio de alcance do aspersor. Com essas informações, qual o número mínimo de aspersores que devem ser ligados para que toda a faixa de grama seja regada.

Regando a grama ("Watering Grass")

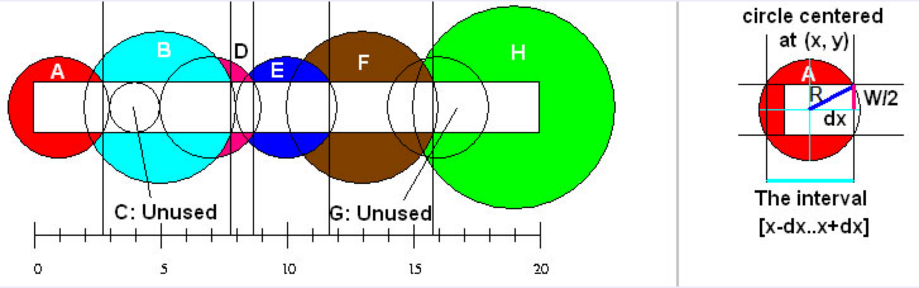


Regando a grama ("Watering Grass")

- Esse problema pode ser transformado em outro problema bem conhecido, denominado problema da cobertura de intervalo ("interval covering problem"), que consiste em descobrir o número mínimo de segmentos que devem ser utilizados para cobrir um certo intervalo.
- Para transformar o problema original no problema de cobertura de intervalos é possível transformar os raios de alcance dos aspersores em segmentos. Supondo um círculo na posição x e raio R, é possível calcular seu segmento de alcance como:

$$[x - dx, x + dx] \qquad dx = \sqrt{R^2 - (W/2)^2}$$

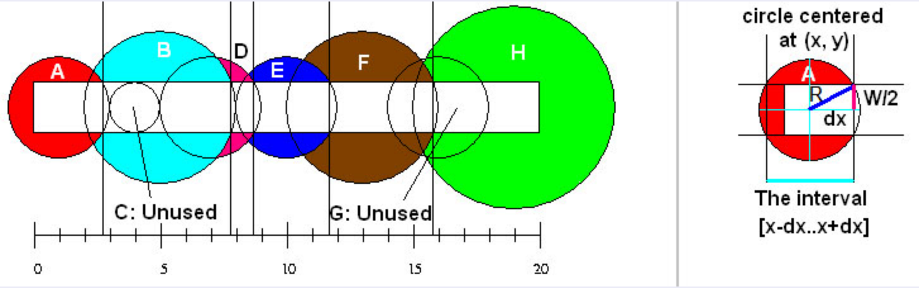
Regando a grama ("Watering Grass")



Regando a grama (“Watering Grass”)

- Agora que o problema original foi transformado no problema de cobertura de intervalos, é possível aplicar o algoritmo guloso já conhecido para esse problema.
- Primeiramente, os segmentos são ordenados de modo crescente pelas posições de suas extremidades esquerdas.
- Considere o intervalo a ser coberto como $[L, R]$. O algoritmo verifica todos os segmentos que contêm L e seleciona o segmento $[a, b_{max}]$ que mais se estende à direita. Em seguida o algoritmo resolve o subproblema $[b_{max}, R]$.

Regando a grama ("Watering Grass")



Referências

1. S. Halim e F. Halim. Competitive Programming 2, Second Edition Lulu (www.lulu.com), 2011. (IMECC – 005.1 H139c)
2. S. S. Skiena, M. A. Revilla. Programming Challenges: The Programming Contest Training Manual, Springer, 2003.
3. T.H. Cormen, C.E. Leiserson, R.L.Rivest e C. Stein. Introduction to Algorithms. 2nd Edition, McGraw-Hill, 2001. (no. chamada IMECC – 005.133 Ar64j 3.ed.)
4. U. Manber. Introduction to Algorithms: A Creative Approach. Addison-Wesley. 1989. (no. chamada IMECC – 005.133 Ec53t 2.ed.)