Aula 10 - Programação Dinâmica

Luís Felipe

UFF

28 de Setembro de 2023

Luís Felipe

28/09/23

Programação Dinâmica

- Até o momento, vimos algumas técnicas de algoritmos bem poderosas, como: divisão e conquista e algoritmos gulosos.
- Vimos também que estas técnicas não podem ser aplicadas a qualquer problema, pois elas só conduzem à resposta ótima em alguns casos específicos. Exemplos?
- Hoje, começaremos a ver uma técnica de aplicabilidade mais geral, que nos ajuda quando técnicas mais especializadas falham.
- Vos apresento: A Programação Dinâmica

Luís Felipe

28/09/23

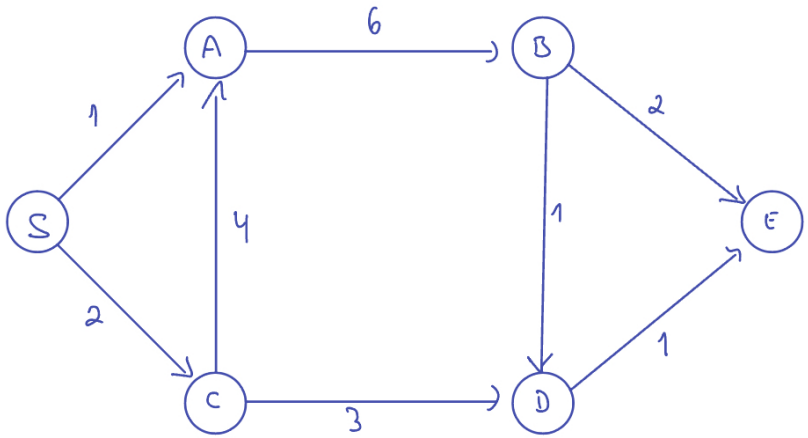
O coração da PD

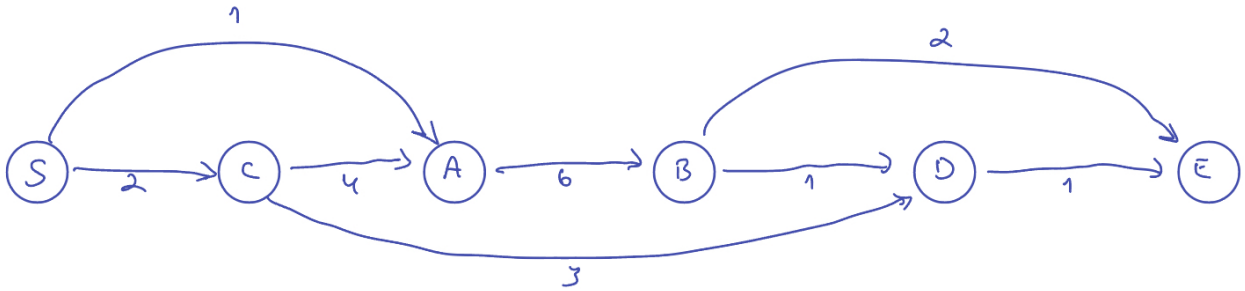
- Pensem num grafo direcionado.
- Sem ciclo (lembram da definição de ciclo neste caso???)
- Um grafo direcionado acíclico (DAG) é onde mora o coração da PD.
- Sabem por quê?
- Um DAG pode ter seus vértices linearizados, i.e., postos em linha com arestas vindo sempre da esquerda para a direita.
- Para melhor exemplificar, vejamos o problema do menor caminho em DAG's.

Luís Felipe

28/09/23

DAG





Luís Felipe

28/09/23

Menor caminho em DAG's

- Dado um DAG, queremos saber qual o menor caminho entre um vértice S e todos os outros.
- A linearização do DAG ajuda muito na hora de calcular essas distâncias.
- Note que, para alcançar o vértice D a partir de S, só temos duas formas: ou por B ou por C.
- Para saber qual é o menor caminho, basta comparar essas duas rotas: $dist(D) = min(dist(B) + 1, dist(C) + 3)$
- Se nós computarmos as distâncias da esquerda pra direita, quando chegarmos no nó do qual queremos a informação da distância, teremos certeza que temos todas as informações que precisamos para calculá-la.
- Então começamos com a menor distância: $d(S) = 0$. Progressivamente vamos resolvendo subproblemas maiores, ou seja, calculando a distância de S para vértices que estão mais adiante na linearização.

Luís Felipe

28/09/23

Um DAG implícito

- A PD é um paradigma algorítmico muito poderoso
- Um problema é resolvido pela solução de uma coleção de subproblemas. Começando do menor
- As respostas obtidas solução a solução ajudam na solução dos problemas maiores
- Até que o problema original seja resolvido
- Um DAG não é dado em PD, ele está implícito em PD. Alguém consegue enxergar o por quê?
- Cada nó é um subproblema e as arestas direcionadas são as dependências entre esses subproblemas.

Luís Felipe

28/09/23

LIS

- O problema da maior subsequência crescente (longest increasing subsequence, LIS) tem como entrada uma sequência de números: $a_1, a_2, \ldots, a_n$.
- Uma subsequência é um subconjunto destes números respeitando a ordem dada pela sequência: $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, onde $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n$.
- Uma subsequência crescente é uma subsequência onde os números vão ficando sempre um pouco maiores.
- O objetivo do problema é encontrar a subsequência crescente de maior tamanho, i.e com mais elementos.

Luís Felipe

28/09/23

LIS

Exemplo: Dada a sequência $5, 2, 8, 6, 3, 6, 9, 7$, qual a LIS?

- $2, 3, 6, 7$
- Neste exemplo, as setas denotam transições entre elementos consecutivos da solução.
- A ideia é, dada uma sequência, criar um grafo com todas as possíveis transições, i.e., para cada número na sequência temos um vértice e teremos a aresta (i, j) para todos os possíveis a_i e a_j que podem aparecer consecutivamente na solução.
  - $i < j$ e
  - $a_i < a_j$
- Esse grafo é um DAG!!
- Então basta encontrarmos o maior caminho neste DAG.

Luís Felipe

28/09/23

Encontrando a LIS...

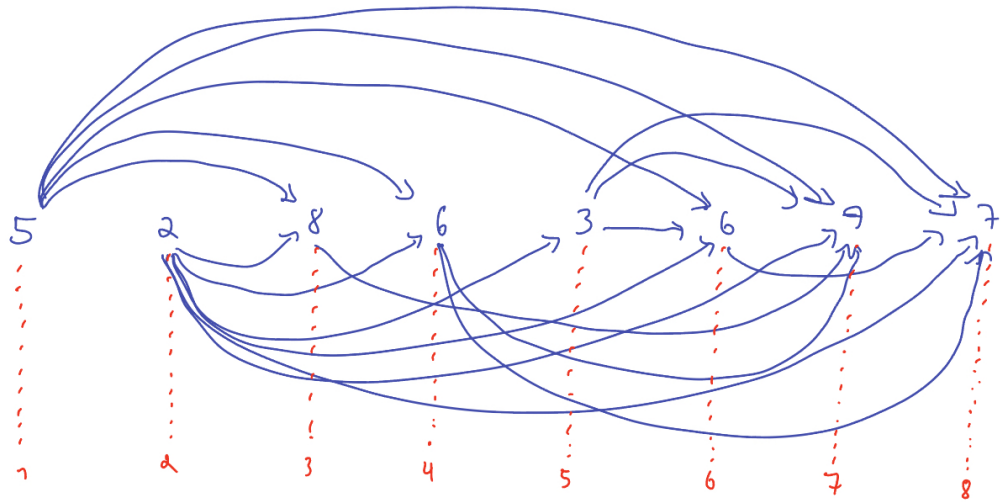
- Seja $L(j)$ o tamanho do maior caminho terminando em j.
- Assim como no problema do menor caminho, qualquer caminho até um determinado nó , tem que passar por um de seus predecessores
- Assim $L(j)$ é $1 + \max\{L(v)\}$, onde v é um predecessor de j.
- Se j não tem aresta de entrada, então $L(j) = 1$. Assim, se $j = 1$, temos que $L(1) = 1$
- Com isso, obtemos os demais $L(v)$
- Seja $G(V, E)$ a DAG construída:

Para $j = 1, \cdots, n$:
  $L(j) = 1 + \max\{L(i) \ : \ (i, j) \in E\}$
Retorne $\max_j\{L(j)\}$

Luís Felipe

28/09/23

Exemplo



$L(1) = 1$ $\quad S(1) = 5$

$L(2) = 1$ $\quad S(2) = 2$

$L(3) = 1 + \max\{L(1), L(2)\} = 2$ $\quad S(3) = 2, 8$

$L(4) = 1 + \max\{L(1), L(2)\} = 2$ $\quad S(4) = 2, 6$

$L(5) = 1 + \max\{L(2)\} = 2$ $\quad S(5) = 2, 3$

$L(6) = 1 + \max\{L(1), L(2), L(5)\} = 3$ $\quad S(6) = 2, 3, 6$

$L(7) = 1 + \max\{L(1), L(2), L(3), L(4), L(5), L(6)\} =$

$= 4$ $\quad S(7) = 2, 3, 6, 9$

$L(8) = 1 + \max\{L(1), L(2), L(4), L(5), L(6)\} =$

$= 4$ $\quad S(8) = 2, 3, 6, 7$

Luís Felipe

28/09/23

Complexidade

- Note encontramos o tamanho da LIS, enquanto para a LIS só irmos armazenando os elementos ao passo que vamos tomando $L(j)$.

- Complexidade: Para cada elemento, precisamos tomar o melhor dos predecessores, ou seja, devemos ao final ter passado por todas as arestas, assim: $O(n^2)$.

- Isso é PD:
  - Para resolver o problema, definimos n subproblemas: $\{L(j) \mid 1 \leq j \leq n\}$
  - Existe uma ordenação dos subproblemas e uma relação que mostra que, resolvidos os subproblemas menores, ou seja, aqueles que aparecem anteriormente na ordenação, sabemos resolver o problema.
  - A relação é a seguinte: $L(j) = 1 + \max(L(i) \ : \ (i,j) \in E)$

- Obs.: Se pensássemos em resolver recursivamente, como seria a solução? Qual seria a complexidade?