Aula 12 - Algoritmo de Bellman-Ford

Luís Felipe

UFF

05 de Maio de 2026

Luís Felipe

05/05/26

Algoritmo de Bellman-Ford

Problema:

- Dado um dígrafo G com pesos nas arestas, determinar caminhos mínimos a partir de um vértice r.
- Agora, permitimos pesos negativos.

Pergunta:

Como adaptar Dijkstra para esse caso?

Resposta:

Algoritmo de Bellman-Ford

Luís Felipe

05/05/26

Ideia do algoritmo

- Diferente de Dijkstra, não usamos abordagem gulosa.
- A ideia é simples:
  - relaxar todas as arestas repetidamente;
- A cada iteração, as estimativas melhoram.
- Após um número suficiente de iterações:

obtemos as distâncias corretas

Inicialização

- Para cada vértice v:

$$L(v) = \infty \quad pai(v) = null$$

- Para a raiz r:

$$L(r) = 0$$

Luís Felipe

05/05/26

Algoritmo de Bellman-Ford

```
1  Entrada: dígrafo G, pesos W, raiz r
2
3  para todo vertice v
4      L(v) <- infinito
5      pai(v) <- null
6
7  L(r) <- 0
8
9  repetir |V| - 1 vezes
10     para cada aresta (u,v)
11         se L(u) + W(u,v) < L(v) entao
12             L(v) <- L(u) + W(u,v)
13             pai(v) <- u
```

Luís Felipe

05/05/26

Algoritmo de Bellman-Ford em Python

```python

def bellman_ford(adj, W, r):

    n = len(adj)

    L = [float('inf')] * n      # L[v] = distancia de r ate v
    pai = [-1] * n              # pai[v] = predecessor de v

    L[r] = 0

    # relaxa todas as arestas |V|-1 vezes
    for _ in range(n - 1):
        for u in range(n):
            for v in adj[u]:
                if L[u] + W[(u,v)] < L[v]:
                    L[v] = L[u] + W[(u,v)]
                    pai[v] = u

    # verifica ciclo negativo
    for u in range(n):
        for v in adj[u]:
            if L[u] + W[(u,v)] < L[v]:
                return None, None, True

    return L, pai, False
```

Por que $|V| - 1$ iterações?

- Suponha, por enquanto, que não há ciclo negativo alcançável a partir de r.
- Nesse caso, para cada vértice v, se existe caminho mínimo de r até v, então existe um caminho mínimo simples, i.e. um caminho simples que não repete vértices.
- Portanto, em um grafo com $|V|$ vértices, ele tem no máximo $|V| - 1$ arestas.
- Após a i-ésima passada por todas as arestas, o algoritmo já considera todos os caminhos de r até cada vértice usando no máximo i arestas.
  - Ou seja, após a 1a passada por todas as arestas já sabemos todos os caminhos ótimos a partir de r que usam no máximo 1 aresta; após a 2a passada, sabemos todos os caminhos ótimos de r com no máximo 2 arestas; e assim sucessivamente.
- Assim, após $|V| - 1$ passadas, todos os caminhos mínimos simples já foram considerados.

Luís Felipe

05/05/26

Intuição das passadas

- Pense em um caminho mínimo de r até v:

$$r = v_0 \to v_1 \to v_2 \to \cdots \to v_k = v.$$

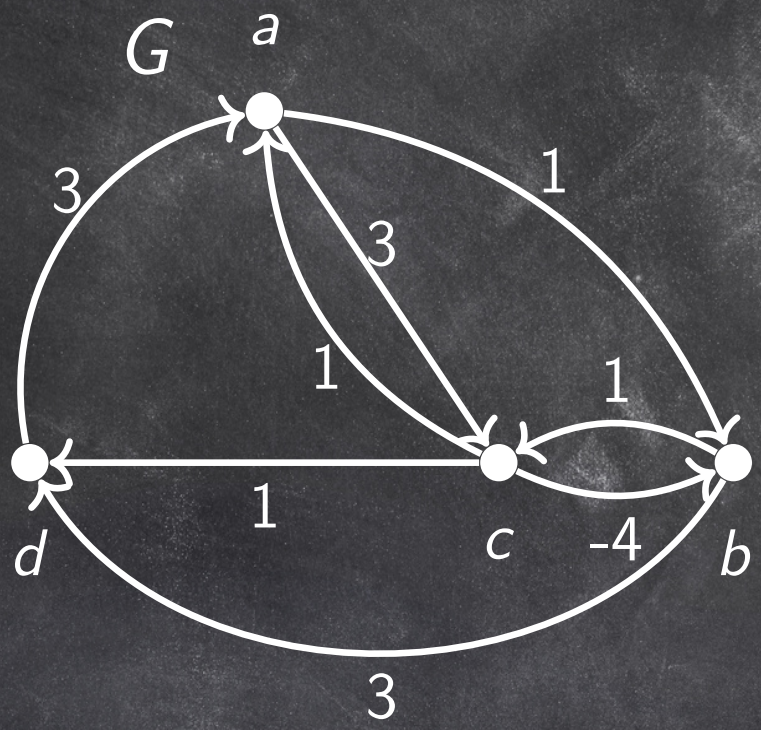
- Depois da primeira passada, conseguimos descobrir corretamente caminhos que usam uma aresta, como $r \to v_1$.
- Depois da segunda passada, conseguimos descobrir caminhos que usam duas arestas, como $r \to v_1 \to v_2$.
- Depois da terceira passada, caminhos com três arestas, e assim por diante.
- Como um caminho mínimo simples tem no máximo $|V| - 1$ arestas, $|V| - 1$ passadas são suficientes.

Luís Felipe

05/05/26

Exemplo

Raiz: a



Luís Felipe

05/05/26

Execução

Passada 1

```
a→b: 0+1=1      → L[b]=1
a→c: 0+3=3      → L[c]=3
b→c: 1+1=2      → L[c]=2
b→d: 1+3=4      → L[d]=4
c→b: 2-4=-2     → L[b]=-2
c→a: 2+1=3      (não melhora)
c→d: 2+1=3      → L[d]=3
d→a: 3+3=6      (não melhora)

L = {a:0, b:-2, c:2, d:3}
```

Passada 2

```
a→b: 0+1=1      (não melhora)
a→c: 0+3=3      (não melhora)
b→c: -2+1=-1    → L[c]=-1
b→d: -2+3=1     → L[d]=1
c→b: -1-4=-5    → L[b]=-5
c→a: -1+1=0     (não melhora)
c→d: -1+1=0     → L[d]=0
d→a: 0+3=3      (não melhora)

L = {a:0, b:-5, c:-1, d:0}
```

Passada 3

```
a→b: 0+1=1      (não melhora)
a→c: 0+3=3      (não melhora)
b→c: -5+1=-4    → L[c]=-4
b→d: -5+3=-2    → L[d]=-2
c→b: -4-4=-8    → L[b]=-8
c→a: -4+1=-3    → L[a]=-3
c→d: -4+1=-3    → L[d]=-3
d→a: -3+3=0     (não melhora)

L = {a:-3, b:-8, c:-4, d:-3}
```

Verificação de ciclo negativo

```
b→c: -8+1=-7 < -4  → ciclo negativo detectado

Ciclo: b → c → b  (peso = -3)
```

Luís Felipe

05/05/26

Corretude do Algoritmo de Bellman-Ford

Invariante: Após a i-ésima passada por todas as arestas, $L(v)$ é no máximo o custo do menor caminho de r até v com até i arestas.

Prova por indução em i:

Base: $i = 0$.

- $L(r) = 0$;
- $L(v) = \infty$ para $v \neq r$;
- isso está correto para caminhos com zero arestas.

Hipótese: Após $i - 1$ passadas, os melhores caminhos com até $i - 1$ arestas já foram considerados.

Passo: Considere um caminho ótimo com até i arestas terminando em v.
Se ele usa i arestas, sua última aresta é alguma aresta (u, v).
Antes dessa última aresta, há um caminho de r até u com até $i - 1$ arestas.
Pela hipótese de indução, esse caminho até u já foi considerado. Ao relaxar (u, v), o algoritmo também considera o caminho até v.

Conclusão: Após $|V| - 1$ passadas, todos os caminhos mínimos simples foram considerados.

Luís Felipe

05/05/26

Detecção de ciclos negativos

- Após $|V| - 1$ passadas, todos os caminhos simples já foram considerados.
- Fazemos então uma passada adicional por todas as arestas.
- Se alguma aresta ainda puder ser relaxada, encontramos um caminho melhor com pelo menos $|V|$ arestas.
- Um caminho com pelo menos $|V|$ arestas repete algum vértice.
- Logo, ele contém um ciclo.
- Se esse ciclo permite diminuir o custo do caminho, então há um ciclo negativo alcançável a partir de r.

Luís Felipe

05/05/26

Complexidade

- Para cada iteração, percorremos todas as arestas.
- Número de iterações: $|V| - 1$
- Logo:

$$O(nm)$$

- Mais lento que Dijkstra, porém mais geral.