

Алгоритми в графи с тегла на ребрата. Оценки за сложност.

Когато говорим за сложност на алгоритми в графи, ще имаме предвид два показателя – броя n на върховете и броя m на ребрата.

Дефиниция: Нека $G(V, E)$ е свързан граф и $c: E \rightarrow R$ е функция с реални стойности, дефинирана по ребрата на графа. Стойността $c(e), e \in E$, наричаме *цена* или *тегло* на реброто e . Нека $D(V, E')$ е покриващо дърво на $G(V, E)$. Цена (тегло) на дървото ще наричаме сумата $c(D) = \sum_{e \in E'} c(e)$. Покриващото дърво $D_0(V, E_0)$ на $G(V, E)$ наричаме *минимално* (*максимално*), ако $c(D_0) \leq c(D)$ (съответно $c(D_0) \geq c(D)$), за всяко друго покриващо дърво D на $G(V, E)$.

Забележка: Понятията максимално и минимално покриващо дърво са аналогични, затова ще говорим за *оптимално покриващо дърво*, означавайки с това понятие кое да е от двете. Ще разглеждаме само минимално покриващо дърво. Аналогично изводи могат да се правят за максимално покриващо дърво, ако се замени $\min$ с $\max$ и се обърнат неравенствата.

Теорема: Нека $G(V, E)$ е свързан граф с ценова функция на ребрата $c: E \rightarrow R$ и $\emptyset \subset U \subset V$. Нека $e = (v_i, v_j) \in E$ е такава, че $v_i \in U, v_j \in V \setminus U$ и e има минимално тегло измежду всички такива ребра. Тогава съществува МПД (минимално покриващо дърво) $D_0(V, E_0)$ на $G(V, E)$, такава че $e \in E_0$.

Доказателство: Допускаме, че не съществува МПД, което да съдържа реброто e . Взимаме си произволно МПД $D(V, E')$. От допускането, $e \notin E'$. Образоваме графа $G'(V, E' \cup \{e\})$. Очевидно G' има точно 1 цикъл, който поне още един път пресича границата м/у $\{U\}$ и $\{V \setminus U\}$. Нека това да бъде реброто (u, v) , но e има минимално тегло измежду всички такива ребра $\Rightarrow c(e) \leq c(u, v)$. Строим дървото $D'(V, E' \cup \{e\} - \{(u, v)\})$, което съдържа e . D' е покриващо дърво на $G(V, E)$ и $c(D') \leq c(D)$. Ако допуснем, че $c(D') < c(D) \Rightarrow$ противоречие с това, че $D(V, E')$ е минимално покриващо дърво.

Сега ще разгледаме едни от най-популярните алгоритми за построяване на МПД, а именно алгоритъма на Прим и алгоритъма на Крускал

Алгоритъм на Прим: Нека е даден $G(V, E)$ и функция $c: E \rightarrow R$, задаваща теглата на ребрата му. Алгоритъмът на Прим строи минимално покриващо дърво с корен зададен връх $r \in V$ на G .

Процедура:

- 1) $D_0(V_0, E_0)$, $V_0 = \{r\}$, $E_0 = \emptyset$ и $k = 0$.
- 2) Нека сме построили $D_k(V_k, E_k)$. Търсим реброто $e = (v_i, v_j)$, $v_i \in V_k$, а $v_j \notin V_k$, което да има минимално тегло и построяваме $D_{k+1}(V_{k+1}, E_{k+1})$, $V_{k+1} = V_k \cup \{v_j\}$, $E_{k+1} = E_k \cup \{e\}$ и $k = k + 1$.
- 3) Ако $V_k = V$ край и обявяваме $D_k(V_k, E_k)$ за оптимално, иначе преминаваме към стъпка 2).

Забележка: Ако за реализацията на алгоритъма използваме матрица на съседства, сложността му ще бъде $O(n^3)$. Ако използваме двоична пирамида и списък на наследниците, сложността може да се сведе до $O(m \log n)$. С използване на пирамида на Фибоначи, можем да ускорим алгоритъма до $O(m + n \log n)$.

Алгоритъм на Крускал: Нека е даден $G(V, E)$ и функция $c: E \rightarrow R$,

задаваща теглата на ребрата му. Алгоритъмът строи минимално покриващо дърво на G (некореново МПД).

Процедура:

- 1) Сортираме ребрата на G в нарастващ ред на цената и нека този ред е $e_0, e_1, \dots, e_m$.
- 2) От всеки връх на графа образуваме тривиално дърво $D_v(\{v\}, \{\})$.
- 3) За всяко ребро $e_i = (v_{i_1}, v_{i_2})$, $i \in I_m$ (по реда определен в сортирането)

правим следното: ако v_{i_1} и v_{i_2} са в различни дървета $D'(V', E')$ и $D''(V'', E'')$ обединяваме двете дървета с реброто e_i $D(V' \cup V'', E' \cup E'' \cup \{e_i\})$.

Забележка: Алгоритъмът на Крускал строи МПД, понеже на всяка стъпка взима най-лекото ребро (понеже обхождаме ребрата от най-леко към най-тежко), т.е. ако едно ребро е по-леко от друго то се избира по-рано за построяване на дървото. Ако двата края на едно ребро са в едно и същ поддърво то второто ребро няма да се вземе (понеже вече има по-леки ребра, които сме взели за свързването на двата края).

Забележка: Алгоритъмът на Крускал е със сложност $O(m \log m)$, което е сложността на сортиране на ребрата. Действително, сложността на стъпка 2 е $O(n)$, а използвайки абстрактния тип „разбиване”, имплементиран с гора от

„повдигани” на всяка find-стъпка коренови дървета, за стъпка 3) ще получим сложност само $O(m \cdot \log^* n)$.

Най-къс път в граф: Нека $G(V, E)$ е свързан граф, а $c: E \rightarrow R^+$ – теглова функция на ребрата с положителни реални стойности. Претеглена дължина на пътя $v_{i_0}, v_{i_1}, \dots, v_{i_l}$ в графа ще наричаме $\sum_{j=0}^{l-1} c(v_{i_j}, v_{i_{j+1}})$. Пътят от v_{i_0} до v_{i_l} с най-малка претеглена дължина наричаме *най-къс път* от v_{i_0} до v_{i_l} . Естествено е да дефинираме претеглена дължина 0 за тривиален път от v до v .

Ще разгледаме следната задача: *Да се намерят дължините на най-късите пътища (и самите най-къси пътища) от зададен връх v_0 до всички останали върхове на свързания граф $G(V, E)$ с теглова функция $c: E \rightarrow R^+$.*

В частния случай $c(e)=1, \forall e \in E$, претеглената дължина е равна на дължината на пътя. За този частен случай решението на задачата се дава от следната:

Теорема: Нека $G(V, E)$ е свързан граф с теглова функция $c(e)=1, \forall e \in E$ и D е покриващо дърво на G корен v_0 , построено в ширина. Пътищата в D от корена v_0 до останалите върхове на G са най-къси пътища от v_0 до тези върхове.

Доказателство: Ще направим индукция по i (нивата на обхождане в ширина).

- 1) $i = 0$ $L_0 = \{v_0\}$ и най-късият път от v_0 до v_0 е тривиалният път с дължина 0.
- 2) Допускаме, че твърдението е вярно за върховете от ниво i .
- 3) Ще докажем, че дължините на най-късите пътища от v_0 до върховете от L_{i+1} са равни на $i+1$ и значи пътищата от корена до тези върхове в дървото са най-къси. Допускаме че $\exists v \in L_{i+1}$, за който дължината на най-късия път от v_0 до v ($v_0, \dots, w, v$) е $k < i+1$. $\Rightarrow$ пътят $v_0, \dots, w$ е най-късия път от v_0 до w и дължината му е $k-1 < i$. Но съгласно индукционното предположение $w \in L_{i-1}$, което е противоречие с това че $v \in L_{i+1}$ (v трябва да е от L_i).

Алгоритъм на Дийкстра: Даден е свързан граф $G(V, E)$ с теглова функция $c: E \rightarrow R^+$ и начален връх $v_0 \in V$. Да се намери дърво на най-късите пътища от v_0 до всички останали върхове от G .

Процедура: Нека $V=\{v_0=0,1,2,\dots,n\}$. ще използваме два масива – dist и part . $\text{dist}[i]$ ще съдържа временно най-късият път от v_0 до i , в $\text{part}[i]$ ще пазим бащата (предшествания връх) на i по този път. На всяка стъпка алгоритъмът ще намира най-късия път до един от върховете, образувайки множеството VISITED , от върхове за които най-късият път е намерен. В началото на алгоритъм $\text{VISITED}=\{0\}$ (само v_0 е посетен и дължината на най-късия път е 0).

- 1) Разширяваме $c: E \rightarrow R^+$ до $c^*: V \times V \rightarrow R^+$, където

$$c^*: V \times V \rightarrow R^+ = \begin{cases} c(v_i, v_j), (v_i, v_j) \in E \\ \text{безкрайност}, (v_i, v_j) \notin E \end{cases}$$

- 2) Нека $\text{dist}[0]=0$; $\text{part}[0]= -1$; $\text{VISITED} = \{0\}$, а $\text{dist}[i]=c^*(0,i)$ и $\text{part}[i]=0$, за $i=1..n$.

- 3) Повтаряме n пъти следните стъпки

- 3.1) Избираме връх $j \notin \text{VISITED}$, за който $\text{dist}[j]$ е минимално и го добавяме към посетените върхове $\text{VISITED} = \text{VISITED} \cup \{j\}$.

- 3.2) За всеки връх $k \notin \text{VISITED}$ пресмятаме

$\text{dist}[k]=\min(\text{dist}[k], \text{dist}[j]+c^*(j,k))$. Ако $\min$ е $\text{dist}[j]+c^*(j,k)$, тогава $\text{part}[k]=j$.

Както се забелязва, алгоритъмът взима първия връх с минимално временно разстояние до началният връх и обявява това разстояние за минимално. Ще докажем, че това поведение на алгоритъма на Дийкстра е основателно. За краткост в следващите твърдения вместо VISITED ще използваме U .

Лема: Върховете на графа влизат в U в ненамаляващ ред на дължината на най-късите си пътища (монотонно свойство)

Доказателство:

- 1) При $U = \{0\}$, $d_0=0$.
- 2) Допускаме че твърдението е вярно за някакво k , т.е. $d_0 \leq d_1 \leq \dots \leq d_k$, където d_i е дължината на най-късия път на върха v_i , влязал в U на i -тата стъпка.
- 3) Допускаме, че $d_0 \leq d_1 \leq \dots \leq d_k > d_{k+1}$. Но $d_{k+1} = \min(d_{k+1}, d_k + c^*(v_k, v_{k+1}))$, а $c^*(v_k, v_{k+1}) > 0 \Rightarrow d_{k+1} < d_k + c^*(k, k+1)$, което е невъзможно, понеже когато сме избирали d_k , то $d_k < d_{k+1} \Rightarrow$ противоречие.

Дефиниция: Нека $G(V, E)$ е граф, $V = \{0, 1, \dots, n\}$. Нека $U \subseteq V$, $U \neq \emptyset$ и $0, i_1, i_2, \dots, i_{k-1}, i_k$ е път от 0 до i_k , такъв че $0, i_1, i_2, \dots, i_{k-1} \in U$. Такъв път наричаме *специален път* по отношение на U .

Теорема: Нека $U \subseteq V$ е получено след поредна стъпка на алгоритъма на Дийкстра, тогава:

- I) $\forall i \in U$ в $\text{dist}[i]$ е дължината на най-късият път от 0 до i (и за $i \neq 0$ в $\text{part}[i]$ е бащата на i по този най-къс път).
- II) $\forall k \notin U$ в $\text{dist}[k]$ е дължината на най-късия специален път (по отношение на U) от 0 до k (в $\text{part}[k]$ е бащата на k по този най-къс път).

Доказателство: Ще докажем паралелно I) и II) с индукция по построяването на U

1) Нека $U = \{0\}$.

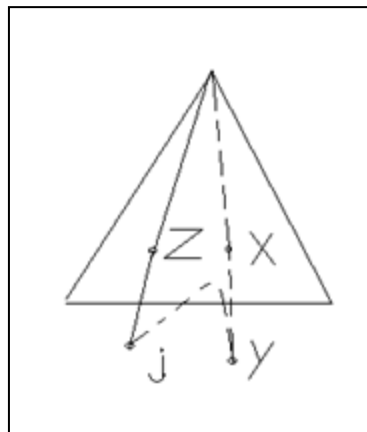
I) Тривиалният път е най-къс път от 0 до 0 (0 е корен на бъдещото дърво на най-късите пътища, следователно няма баща).

II) Специален път до k е само реброто $(0, k)$, тъй като $U = \{0\}$. В действителност, в този момент $\text{dist}[k]$ съдържа дължината на съответното ребро (безкрайност за тези, които не са свързани с 0). Освен това за всеки връх k в $\text{part}[k]$ е посочен като баща 0, което съответства на така получените временно най-къси пътища.

2) Да допуснем че твърденията I) и II) са в сила за някое U .

3) Нека $U' = U \cup \{j\}$

I) За $\forall i \in U$ в $\text{dist}[i]$ е дължината на най-късия път съгласно индукционното предположение I). Да допуснем, че запомненето в $\text{dist}[j]$ не е дължина на най-къс път от 0 до j (съответно запомненият път в part не е най-къс). Но съгласно индукционното предположение II) това е най-късият специален път до j по отношение на U . Откъдето следва, че най-късият път не е специален и поне веднъж напуска U преди да стигне до j . Нека да означим с y първият връх $\notin U$, по този път, а с z – предпоследния връх по запомнения в part път. От това че специалният път $0, \dots, z, j$ не е най-къс следва, че



следва, че $\text{dist}[j] = c^*(0, \dots, z, j) > c^*(0, \dots, x, y, \dots, j) > c^*(0, \dots, x, y)$, Второто неравенство идва от това че ребрата са неотрицателни. Но $0, \dots, x, y$ е най-къс специален път до $y \Rightarrow \text{dist}[y] =$

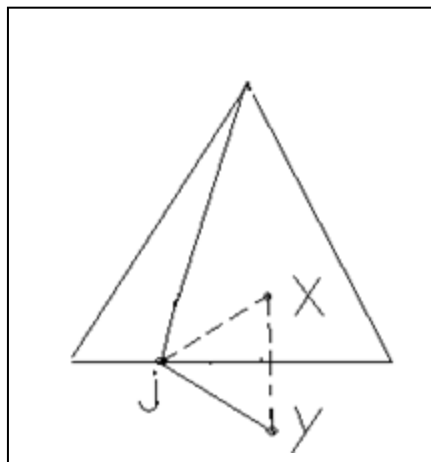
$c^*(0, \dots, x, y) < \text{dist}[j]$, но това е противоречие с избора на j . Алгоритъмът на Дийкстра трябваше да избере y . Следователно допускането е невярно и специалният път до j е глобално най-къс.

II) Съществуват 3 възможности за $k \notin U$ след разширяването на U с j :

а) $\text{dist}[k]$ да остане най-къс специален път до k и спрямо U' – алгоритъмът отчита тази възможност.

б) $\text{dist}[k] = \text{dist}[j] + c^*(j, k)$ да стане най-къс специален път до k спрямо U' . Алгоритъмът отчита и тази възможност.

в) най-късият специален път до k да минава през j , но j не е предпоследен връх. Нека предпоследният връх по най-късия специален път до върха j е x от U (т.е. x е влязал в U преди j). $\Rightarrow c^*(0, \dots, j) < c^*(0, \dots, j, \dots, x)$, защото теглата на ребрата са положителни. Но това е в противоречие с Лемата за монотонност.



Няма други случаи, понеже индукционното предположение изключва възможността да има минимален специален път, използващ j , различен от запомнения

Забележка: Алгоритъмът има сложност $O(n^2 + m) = O(n^2)$, Понеже за всеки връх се опитваме да намерим най-краткият път до останалите непосетени. Такава сложност ще получим, ако пазим дължините на временно най-късите пътища в масив. Но ако пазим най-късите пътища в приоритетна опашка и представим графа със списък на съседите, можем да получим алгоритъм със сложност $O((n + m) \log n) = O(m \log n)$ – добра при графи с малко ребра. Но най-добра сложност се постига с използване на пирамида на Фибоначи – $O(m + n \log n)$.

Алгоритъм на Флойд: Нека е даден граф $G(V, E)$, алгоритъмът намира най-кратките разстояния между всяка двойка върхове от графа и то без да е необходима допълнителна памет. Ще използваме с матрица на теглата $A[i][j]$ на графа G . В началото в $A[i][j]$ стои стойността на реброто (i, j) , ако има такова или ∞ в противен случай. Нека $n = |V|$.

Процедура:

```
for (k = 1; k ≤ n; k++) {  
    for (i = 1; i ≤ n; i++) {  
        for (j=1; j ≤ n; j++) {  
            if (A[i][j] > A[i][k] + A[k][j]) {  
                A[i][j] = A[i][k] + A[k][j]  
            }  
        }  
    }  
}
```

Забележка: Алгоритъмът започва с реброто между m/u i и j , като временно най-къс път – най-къс път не минаващ през други върхове. Да допуснем че знаем дължините на най-късите пътища от всеки връх до всеки друг връх, не минаващи през върховете $k, k+1, \dots, n$. Тогава на поредната k -та стъпка алгоритъмът проверява само дали пътят през върха k не е по-къс от текущ намереният и ако е така, запомня неговата дължина. Алгоритъмът е със сложност $\Theta(n^3)$, понеже има 3 вложени цикъла.