ДАА семинар 12

Нач-кви пътища в графи

- SSSP

Алгоритми, които ще използваме:

- Dijkstra - използва приоритетна опашка от реда
  - Работи само за неотрицателни тегла!
  - Подобен на алг. на Prim
  - $T(n,m) = n.T_{EM} + m.T_{DK} = \dots$

    Extract Min — Decrase Key

1) $= n\,O(\lg n) + m\,O(\lg n) = O((n+m)\lg n)$ → Binary Heap

2) $= n.O(\lg n) + m.O(1) = O(n\lg n + m)$ → Fibonacci Heap

// За обща култура. Fibonacci Heap не се разглежда в курса.

- Bellman-Ford
  - Работи за отрицателни тегла, но без отрицателни цикли.

    ↓

    Ако има отрицателни цикли, ги намира. (и връща False) например
  - $T(n,m) = O(n.m)$

Зад. За специални видове графи има и по-оптимални алг.

Например за DAG има линеен по време.

Зад Има n студенти на изпит. Знаем кои могат да си подсказват, като за всяка двойка знаем дали това е опасно или не. За дадени двама студенти да се намери най-безопасният път за подсказване.

Решение: Постр. граф с върхове съответстващи на всички студенти.

Ребро м/у два върха ще ТСТК съответстващите студенти могат да си подсказват, с тегло 0, ако това е безопасно и 1 иначе.

Търсим най-кратък път от u до v, съответно за двамата студенти, с Дейкстра (например)

Зад Даден е град с n сгради, k от които са болници, останалите - къщи. ($k < n$). Знаем кои сгради с кои други е свързана и колко е дълъг пътят. Да се намери коя къща е най-отдалечена от болница.

Решение: Искаме да намерим къщата, за която най-късото разстояние до болница (която и да е) е максимално.

Нека имаме граф с n върха - сградите и m ребра - директните връзки м/у тях.

//$\Theta(n+m)$ доп. памет за построяване

Ин. От всяка къща пускаме Dijkstra до всяка болница

$T(n,m,k) = (n-k)\,O(n \lg n + m) = O(n^2 \lg n - kn \lg n + mn - km) =$

$= O(n^2 \lg n + mn)$

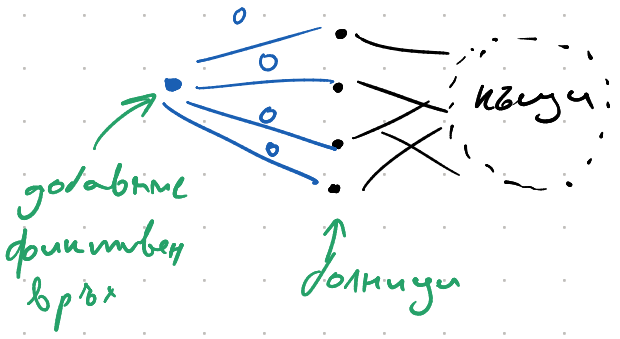
($n-k$ - брой къщи; Dijkstra)

II н. От вс. долница до всяка къща Dijkstra.

$T(n,m,k) = k.\ O(n \lg n + m) + O(n-k) = O(kn \lg n + km + n - k)$

ф. долници — Dijkstra — намиране на max измежду къщите.

III н. Можем да добавим един фиктивен връх:



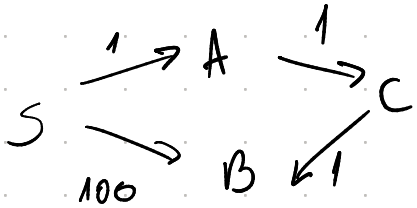
За всяка къща най-кас път до фиктивния връх е най-кас път до долница.

$T(n,m,k) = O((n+1)\lg(n+1) + (m+k)) + O(n-k) =$

$= O(n \lg n + m + \dots)$

Зад. 3 Подготвяйки лекцията си по ДАА за алгоритъма на Дийкстра, доцент Алгоритмов получава блестяща идея: да направи алгоритъма на Дийкстра хем по-бърз, хем по-лесен за имплементиране, като замени приоритетната опашка с обикновена опашка FIFO (First-In, First-Out). Какво бихте казали за тази идея? (поправка 30.08.2022)

Решение:



Ако търсим път от S до B:
От опашката
1. извадихме S ⇒ d(A) = 1
d(B) = 100
2. извадихме B ⇒ без промяна
3. изв. C ⇒ без промяна
4. изв. A ⇒ d(C) = 2 и пр. без промяна

⇒ d(A) = 1 ✓ d(B) = 100 ½ d(C) = 2 ✓

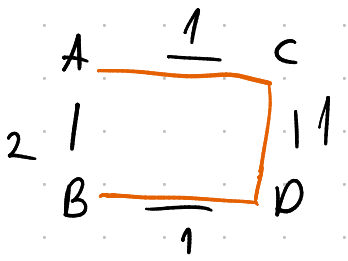
// Много не ми задължава да изведем А преди С и да получим правилен резултат.

Изпит 27.06.2015:

15 т. **Зад. 4** Докажете или опровергайте, че ва всеки свързан тегловен неориентиран граф $G = (V, E)$ с тегловна функция w е вярно, че за всеки два върха $u, v \in V$ е изпълнено $\text{dist}_T(u, v) = \text{dist}_G(u, v)$, където

- T е графът, който връща Алгоритъмът на Крускал с вход (G, w).
- $\text{dist}_T(u, v)$ е най-малката претеглена дължина на път между u и v в T
- $\text{dist}_G(u, v)$ е най-малката претеглена дължина на път между u и v в G
- *претеглена дължина на път* p в някакъв тегловен граф е сумата от теглата на ребрата на p.

Решение:



– МПД

$dist_T(A,B) = 3$

$dist_G(A,B) = 2$

Зад (DP) Ако в тегловен граф с неотриц. тегла всички тегла се увеличат с неотриц. константа:

- Ще се запази ли МПД? (като множество от ребра)
- Ще се запазят ли най-късите пътища? (като последователност от върхове)

Динамично програмиране

- когато имаме общи подзадачи (общи сметки)
- оптимално решение чрез оптимални подрешения
- при комбинаторни задачи

- Memoization → top-down

  DP → bottom-up

Пример за ДП - общи стъпки:

Числа на Фибоначи:

Рекурсивно:

```
fib(n)
1  if n=0
2    return 0
3  if n=1
4    return 1
5  return fib(n-1) + fib(n-2)
```

Итеративно - чрез DP:

```
fib(n):
1   res[1..n] ← нулеви масив
2   res[1] ← 1
3   res[2] ← 1
4   for i ← 3 to n
5     res[i] ← res[i-1] + res[i-2]
6   return res[n]
```

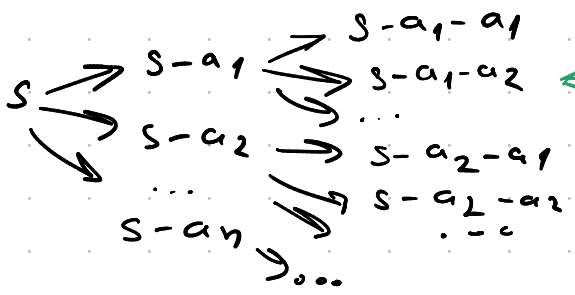
Зад Разполагаме с неограничен брой монети с номинали съответно записани в $A[1..n]$. Да се намери с колко най-малко монети можем да съберем сума S.

Пр $S=20$ $A=[16, 1, 10]$

алчен подход ⇒ 16 + 1 + 1 + 1 + 1

оптимално ⇒ 10 + 10

Решение:



Може направо да пишем код, но рекурсивната декомпозиция е важна за доказване на коректността!

$$f(x) = \min \begin{cases} 1 + f(x - A[1]) \\ 1 + f(x - A[2]) \\ \dots \\ 1 + f(x - A[n]) \end{cases} = 1 + \min_{i=1..n} f(x - A[i])$$

$f(x) = +\infty$ за $x < 0$

1. $M[0 \dots S]$ // всяка попълнена стойност на индекс x съдържа $f(x)$, т.е. min брой монети за сума x
2. $M[0] \leftarrow 0$
3. for $x \leftarrow 1$ to S
4. $\quad M[x] \leftarrow +\infty$
5. $\quad$ for $i \leftarrow 1$ to n
6. $\quad\quad$ if $A[i] < x$
7. $\quad\quad\quad M[x] \leftarrow \min(M[x], M[x - A[i]])$
8. return $M[S]$

$T(n,S) = \Theta(n.S)$ → псевдополиномиална

Зад. Дадени са два символни низа $S_1[1..n]$ и $S_2[1..m]$. Да се намери дължината на най-дългата обща подредица на S_1 и S_2.

// Тази задача е пример за припокриващи се подструктури

Решение: Нека $f(i,j)$ е дължината на най-дългата търсена поредица за $S_1[1..i]$ и $S_2[1..j]$

$$f(i,j) = \begin{cases} 0 & \text{, ако } i=j=0 \\ f(i-1,j-1)+1 & \text{, ако } S_1[i] = S_2[j] \text{ и } i,j>0 \\ \max(f(i,j-1), f(i-1,j)) & \text{, ако } i,j>0 \text{ и } S_1[i] \neq S_2[j] \end{cases}$$

Рекурсивно пресмятане → експоненциална сложност

Итеративно чрез ДП:

```
Alg LCS(S1[1..n], S2[1..m])
1    dp[0..n][0...m] ← празна матрица
2    for i←0 to n
3        dp[i][0] = 0
4    for j←0 to m
5        dp[0][j] = 0
6    for i←1 to n
7        for j←1 to m
8            if S1[i] = S2[j]
9                dp[i][j] = dp[i-1][j-1] + 1
10           else
11               dp[i][j] = max(dp[i-1][j], dp[i][j-1])
12   return dp[n][m]
```

// Ако има нетипична наредба: низове, масиви от числа, коренови дървета и пр. → ДП

- Още един алгоритъм за най-къс път в граф:

  Ако търсим най-къс път от и до v за $\forall u \forall v \in V$,

  то може да използваме алг. на

  Floyd - Warshall с $T(n) = \Theta(n^3)$

// вместо $\Theta(n^4)$ за n-кратно пускане на Bellman-Ford