Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

Video na Youtube

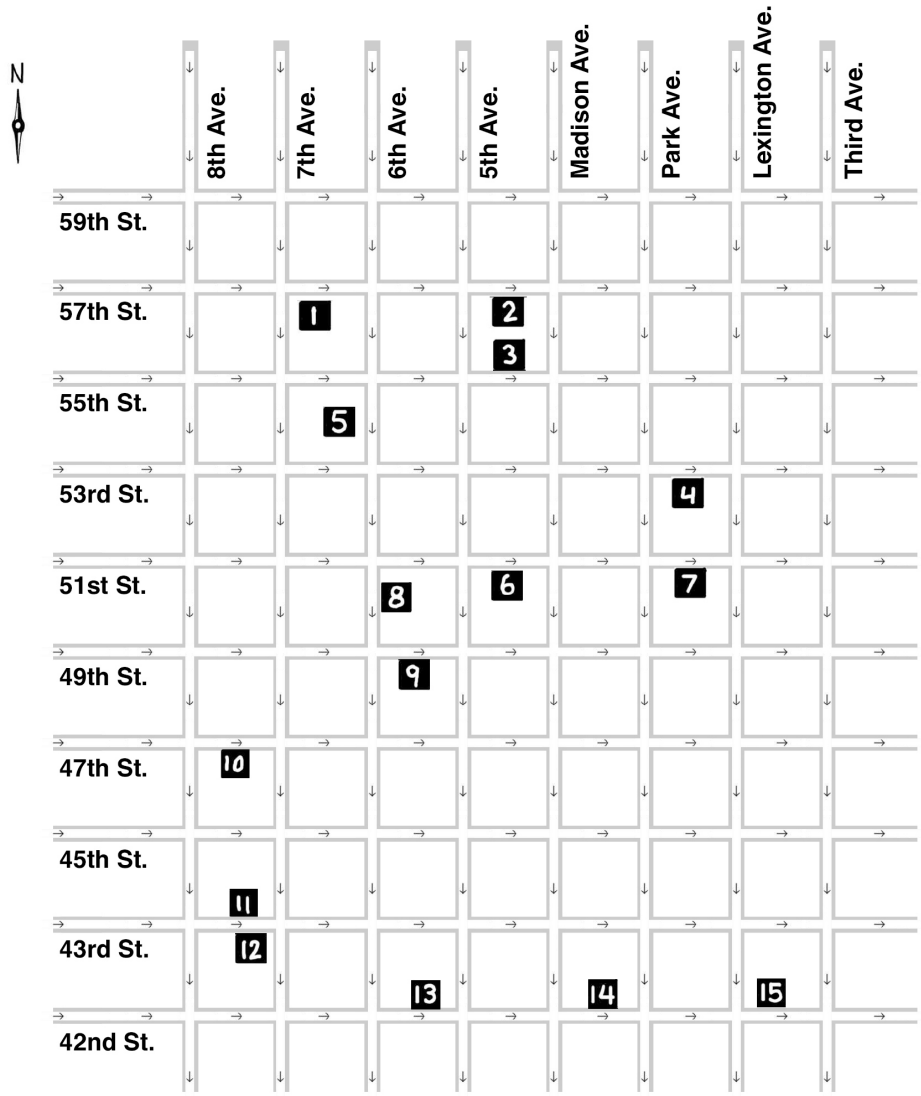
Dynamické programování

Úloha turisty na Manhattanu (*Manhattan tourist problem*)

- máme turistu v levém horním rohu Manhattanu
- turista chce za jeden den projít co nejvíce zajímavých míst, ale co nejméně se nachodit
- řekne si tedy, že se bude pohybovat jen směrem na jih a na východ
- úkolem je najít trasu, na které projde co nejvíce zajímavých míst

Tomáš
Oberhuber

Dynamické programování

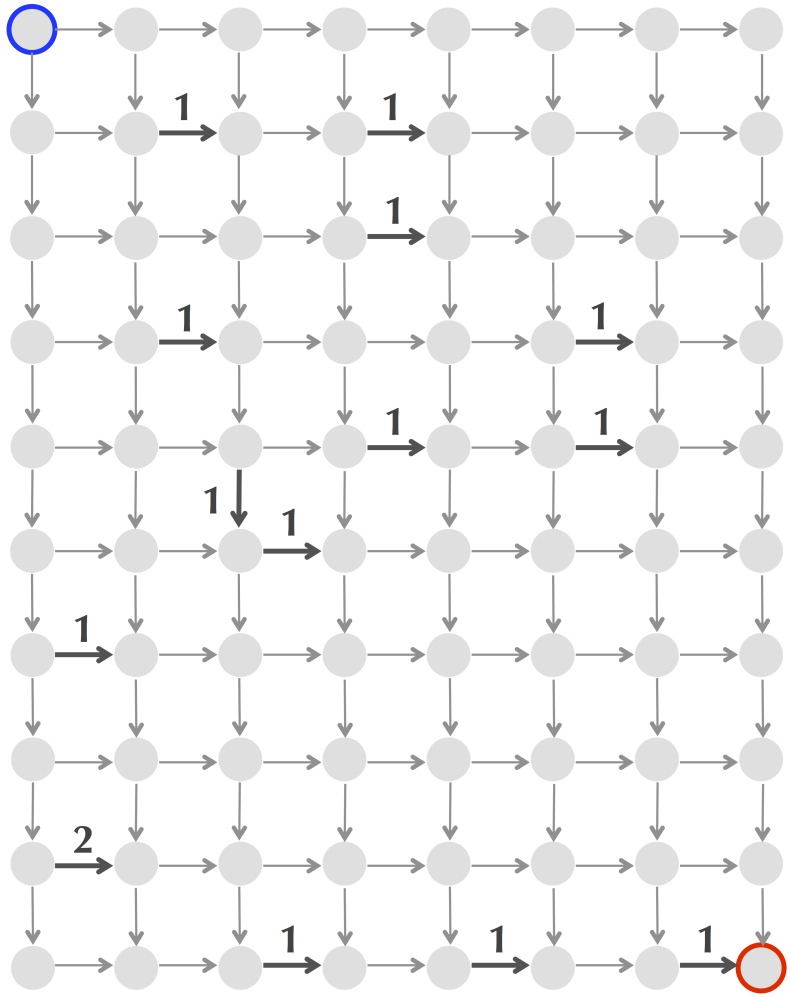


Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování

- vytvoříme si nejprve graf, kde křižovatky tvoří uzly a ulice hrany
- každou hranu ohodnotíme podle toho, jak turisticky zajímavé atrakce tam jsou

Dynamické programování

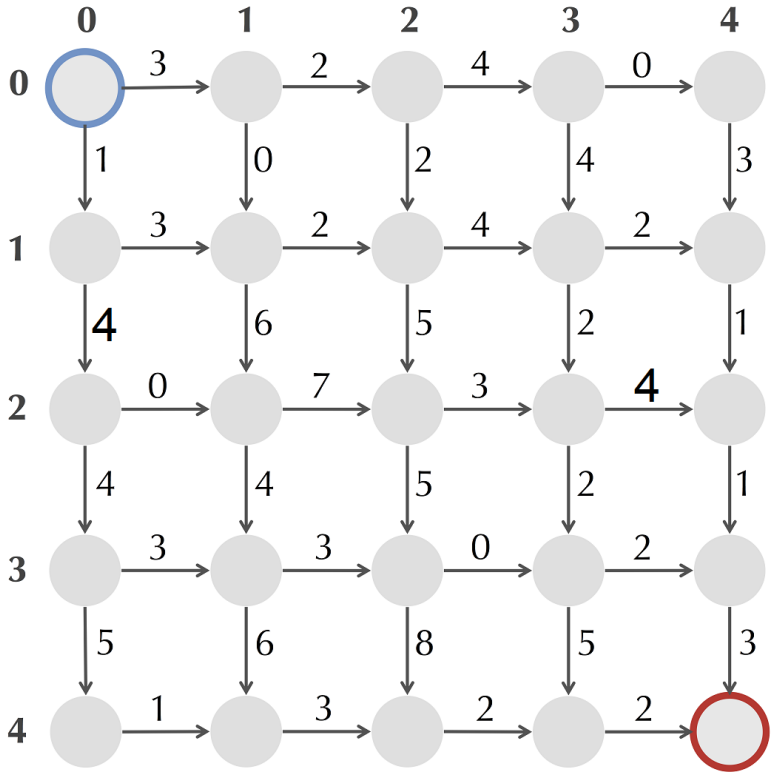


Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování

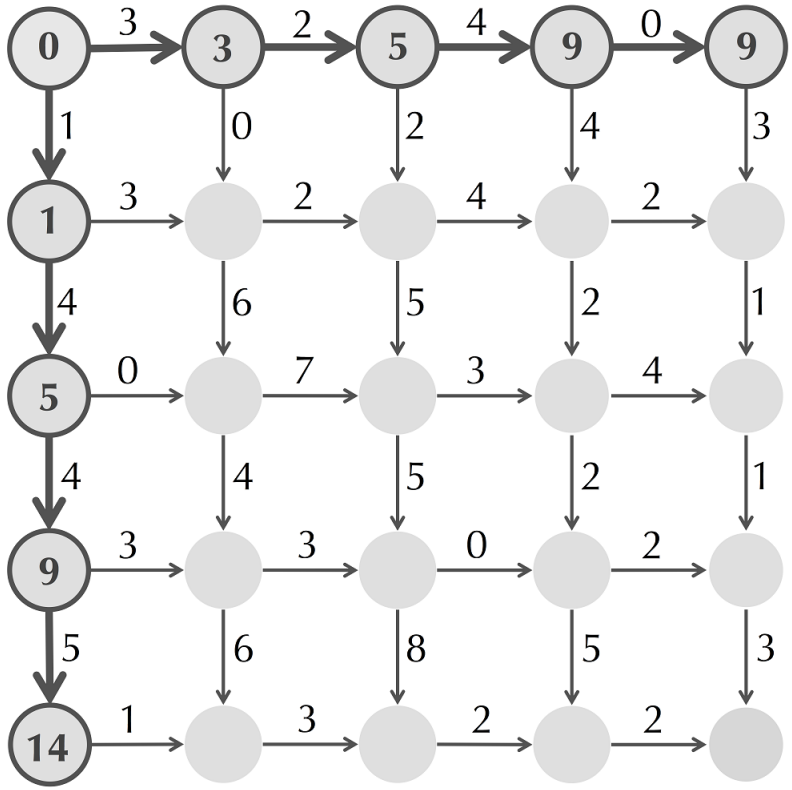
- nyní už jde o nalezení nejdelší cesty v grafu z levého horního rohu do pravého dolního
- úlohu řešíme algoritmem "vlna", který známe ze ZALGu
- postupně do každého vrcholu vyplňujeme délku nejdelší cesty z levého horního rohu do daného vrcholu
- k napočítání nejdelší cesty do určitého vrcholu potřebujeme znát nejdelší cestu do jeho levého a horního souseda
- přitom nejdelší cesty podél horní a levé hranice grafu lze napočítat snadno, tím také začneme

Dynamické programování



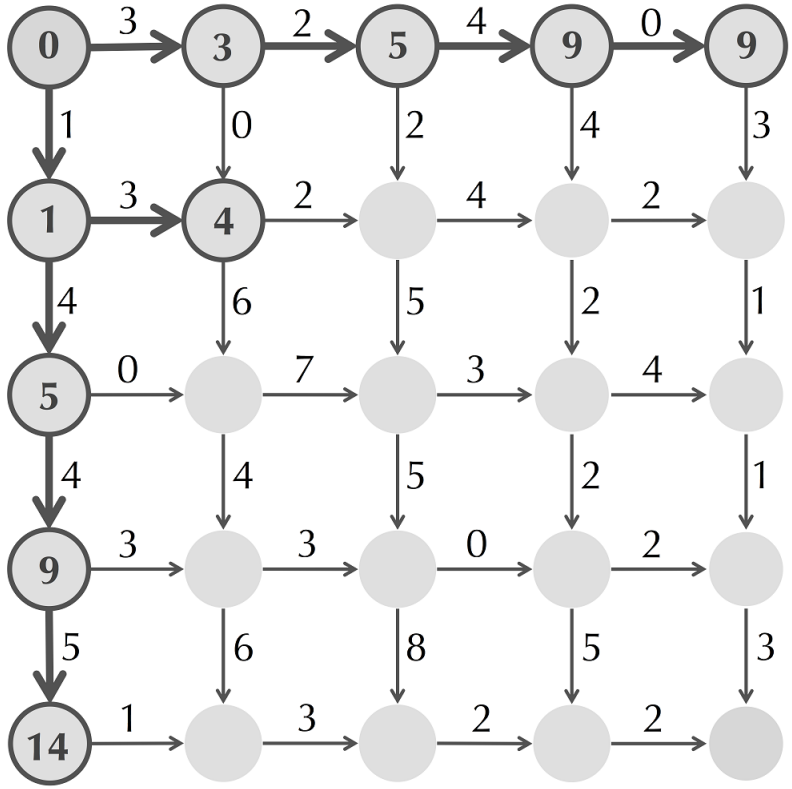
Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování



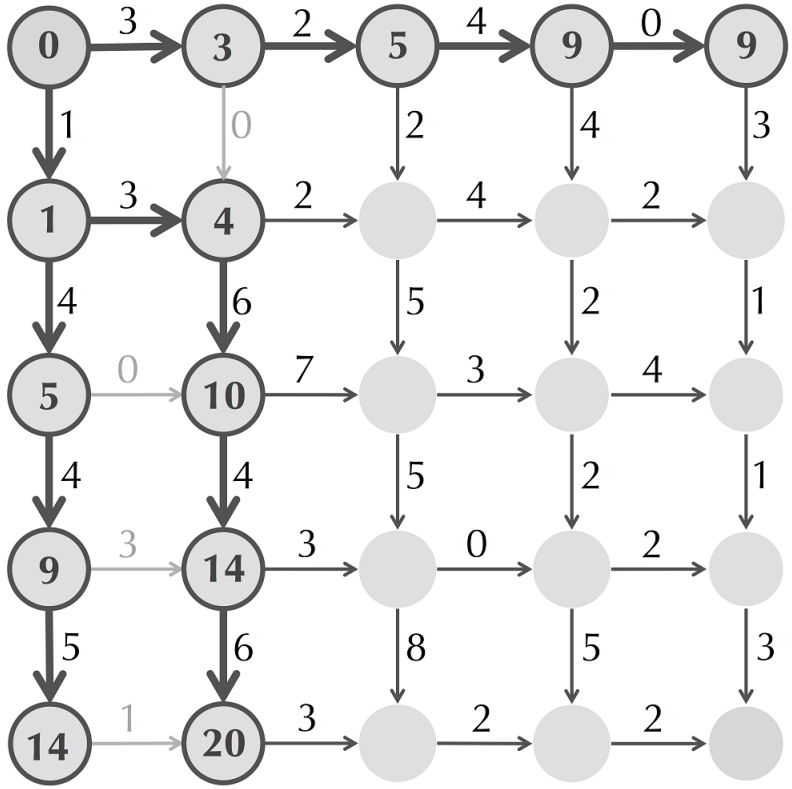
Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování



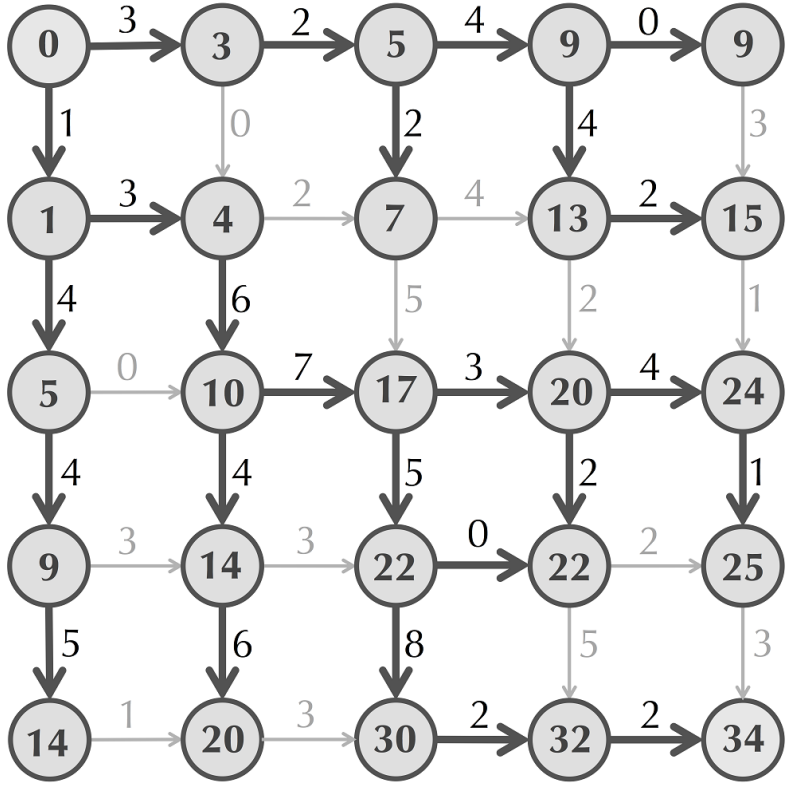
Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování



Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování



Zdroj: http://www.ynegve.info/Post/192/manhattan-tourist-problem

Dynamické programování

- označme si uzlu grafu jako s_{ij} pro $i = 1, \ldots, m$, $j = 1, \ldots, n$
- váhy hran jdoucích dolu označíme jako $w^{\downarrow}$
- váhy hran jdoucích doprava označíme jako $w^{\rightarrow}$
- na začátku definujeme $s_{11} := 0$
- dále počítáma
- pro $i = 1$ a $j = 2, \ldots, n$

$$s_{1,j} := s_{1,j-1} + w^{\rightarrow}_{1,j-1}$$

- pro $j = 1$ a $i = 2, \ldots, m$

$$s_{i,1} := s_{i-1,1} + w^{\downarrow}_{i-1,1}$$

- pro $i = 2, \ldots, m$, $j = 2, \ldots, n$

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + w^{\downarrow}_{i-1,j} \\ s_{i,j-i} + w^{\rightarrow}_{i,j-1} \end{cases}$$

Dynamické programování

- algoritmus lze zapsat takto

1: **procedure** MANHATTANTOURIST(s, $w^{\rightarrow}$, $w^{\downarrow}$, m, n)
2: $s_{11} := 0$
3: **for** $i := 2$ **to** m **do**
4: $s_{i,1} := s_{i-1,1} + w^{\downarrow}_{i-1,1}$
5: **end for**
6: **for** $j := 2$ **to** n **do**
7: $s_{1,j} := s_{1,j-1} + w^{\rightarrow}_{1,j-1}$
8: **end for**
9: **for** $i := 2$ **to** m **do**
10: **for** $j := 2$ **to** n **do**
11: $$s_{ij} := \max \begin{cases} s_{i-1,j} + w^{\downarrow}_{i-1,j} \\ s_{i,j-i} + w^{\rightarrow}_{i,j-1} \end{cases}$$
12: **end for**
13: **end for**
14: return s_{mn}
15: **end procedure**

Dynamické programování

- pokud chci cestu následně zrekonstruovat, mám dvě možnosti
  - u každého uzlu si v průběhu výpočtu pamatovat, odkud jsem do něj přišel
  - jdu od pravého dolního rohu a postupně v každém uzlu hledám souseda, jehož hodnota se liší právě o váhu hrany vedoucí z tohoto souseda