DSA zadanie 3 Popolvár

Peter Plevko

FIIT Stu

Znenie zadania

Našou úlohou v rámci tohto zadania bolo implementovať funkciu
int *zachran_princezne(char **mapa, int n, int m, int t, int *dlzka_cesty). Táto funkcia má nájsť cestu, ktorou postupovať aby Popolvár čo najskôr zneškodnil draka a následne zachránil všetky unesené princezné. Toto zadanie sme mali riešiť pomocou minimálnej haldy a dijkstrovho algoritmu.

Uvedenie do tematiky:

Binárna halda

Binárna halda je dátová štruktúra ktorá si berie formu binárneho stromu. Binárna halda je bežný spôsob ako implementovať prioritne rady.

Binárna halda je definovaná ako binárny strom s dvomi ďalšími obmedzeniami

1 Tvar: binárna halda je kompletný binárny strom to znamená všetky úrovne stromu okrem najhlbšej sú plne naplnene a ak posledný level nie je naplnený tak sa plní z lava doprava.

2 Vlastnosť: Každý kľuč uložený v uzle je väčší alebo rovný ako v deťoch(max heap), Každý kľuč uložený v uzle je menší alebo rovný ako v deťoch (min-heap).

Operácie :

1 vloženie elementu

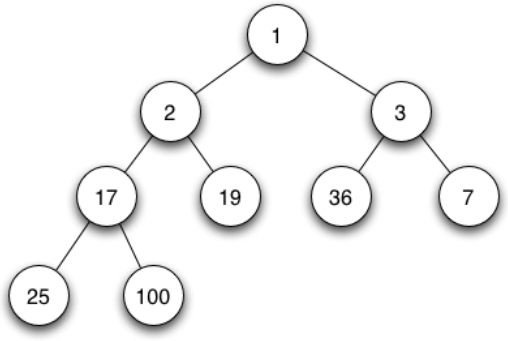
2 vybratie najmenšie/najväčšieho elementu záleží na type

3 usporiadanie

Časová zložitosť operácie v binárnej halde

Algoritmus	Priemer	Najhorší prípad
Vloženie	O(1)	O(n)
Zmazanie	O(log n)	O(log n)

Zložitosť v mojom prípade: O(|V|+|E|*log(|V|))



Príklad minimálnej haldy

Dijkstrov algoritmus

Je to algoritmus na hľadanie najkratších ciest medzi dvomi uzlami v grafe.

Tento algoritmus existuje vo veľa variantoch Dijkstrov originálny algoritmus našiel najkratšiu cestu medzi dvomi danými uzlami, ale bežnejšia metóda urči jeden uzol ako štart a hľadá najkratšie cesty do všetkých ostatných vrcholov. Môže tiež byt použitý na nájdenie najkratšej cesty z bodu a do bodu b stačí tento algoritmus zastaviť v momente keď sme tuto najkratšiu cestu našli.

Dijkstrov algoritmus používa iba kladne čísla respektíve kladne ohodnotenia hrán ktoré sú úplne usporiadane.

Dijkstra je greedy algoritmus v našom prípade to znamená že najprv nájde draka tam nie je žiaden problém ale z draka hľadá najbližšiu princeznú vo vzdialenosti od draka následne od princeznej k ďalšej najbližšej princeznej, avšak môžu nastáť situácie kde keď takto postupujeme celkove ohodnotenie je horšie oproti ohodnoteniu kde vyskúšame všetky kombinácie.

Worst case O(|E|+|V|log|V|)

Popis algoritmu

1. Nastavíme všetky vrcholy na nenavštívene a vytvoríme pole nenavštívených vrcholov

2. Nastavíme každý vrchol na nejakú vzdialenosť to znamená že nášmu štartovnému vrcholu nastavíme 0 a všetkým ostatným nastavíme nekonečno.

3. Pre terajší vrchol skontrolujeme všetkých nenavštívených susedov a vypočítame ich vzdialenosti prístupne cez tento vrchol ak sme našli efektívnejšiu cestu prepíšeme vzdialenosť tohto bodu.

4. keď sme hotový a navštívili sme všetky dostupne vrcholy s daného vrcholu nastavíme tento vrchol na navštívený a odstránime ho z poľa nenavštívených vrcholov.

5. Ak daný vrchol je navštívený alebo najmenšia vzdialenosť medzi vrcholmi v poli nenavštívených vrcholov je nekonečno to znamená že k tomuto bodu sa nedá dostať, skončí algoritmus vykonal svoju prácu

6. Nastav nenavštívený vrchol s najmenšou vzdialenosťou ako daný vrchol a vráť sa na krok 3

Zložitosť v mojom prípade:

P- počet princezien (max. 5)

O(P!+(p^2)*(|E|+|V|)*log(|V|))

Popis môjho algoritmu

Môj algoritmus sa začne zavolaním funkcie zachráň princezné ktorej poskytneme potrebne parametre.

Na začiatku si z mapy zadanej užívateľom vytvorím graf respektíve dvojrozmerne pole štruktúr ktoré ma všetky atribúty ktoré potrebujem. Následne sa zavolá funkcia zatep_draka v ktorej sa zavolá funkcia grapt_to_heap tato funkcia mi nastaví na pozíciu 0 v heape súradnice [0][0] a následne nahádže do heapu všetky ostatne súradnice. V tejto funkcii používam minimálnu haldu z ktorej si extrahujem najmenší prvok respektíve prvý a s tohto prvku hľadám susedov ktorým následne nastavím vzdialenosti pomocou dijkstrovho algoritmu. Tato funkcia mi navráti informácie o bode kde sa nachádza drak.

Po dokončení funkcie zatep_draka pokračujeme v kóde mojou ďalšou funkciou je create_path tato funkcia len prepise cestu k tomuto drakovi od bodu[0][0] do jednorozmerného poľa. V tejto funkcii zachráň princezné samozrejme ošetrujem okrajove situácie ale tie sú zrozumiteľné tak o nich nebudem písať. Vytvorím si pole pr_arr do ktorého uložím na nultú pozíciu struct s informáciami o drakovi a jeho graf. Po zavolaní funkcie find_princeses mam v pr_array uložene na pozícii 0 draka a všetky informácie o ňom následne na ostatných pozíciách mam všetky informácie o princeznách. Po zavolaní funkcie princess_paths si načítam do pr_array načítam cestu od draka ku každej princeznej, a od každej princeznej ku ostatným princeznám.

V tejto chvíli už mame skoro vyhraté stačí už len zistiť v akom poradí mam tieto princezné vyzdvihnúť nato mi slúži funkcia princess_order ktorá zisti najefektívnejšiu cestu od každej princeznej ku každej princeznej pomocou dijskru a následne pomocou permutácii zistím celkovú najefektívnejšiu cestu všetko čo teraz treba už je len toto poradie princezien pretvoriť do výslednej cesty nato slúži funkcia add_pr_path. V tejto chvíli mam všetko čo potrebujem. Už len uvoľním grafy a prepíšem moju cestu do popolvár formátu.

Test 1 defaultny

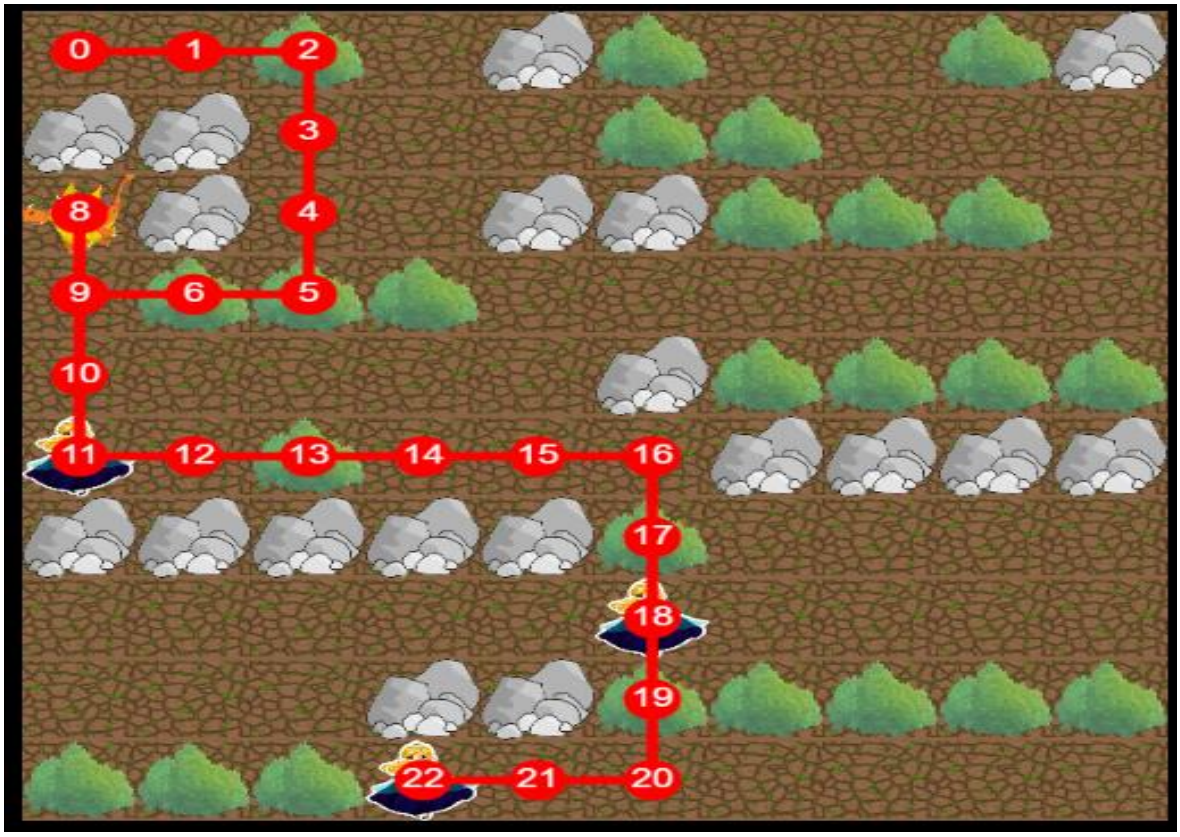
Vstup

```
10 10 12

CCHCNHCCHN
NNCCCHHCCC
DNCCNNHHHC
CHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHH
HHHPCCCCCC
```

Výstup

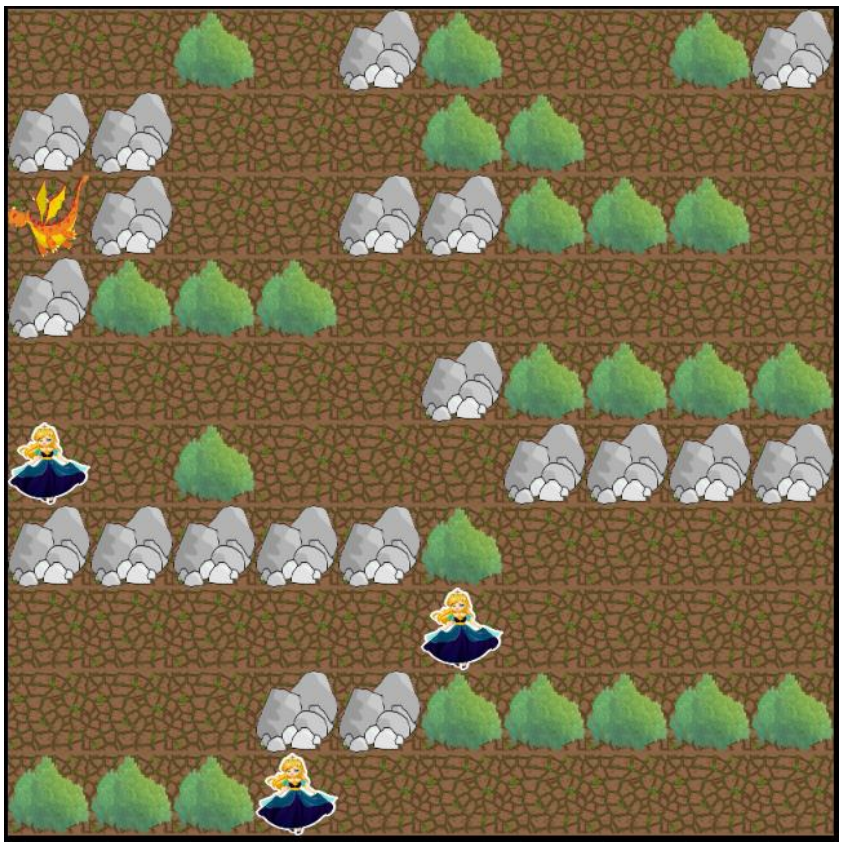
```
0 0
1 0
2 0
2 1
2 2
2 3
1 3
0 3
0 2
0 3
0 4
0 5
1 5
2 5
3 5
4 5
5 5
5 6
5 7
5 8
5 9
4 9
3 9
Čas: 29
```



Test 2 nedostupný drak

Vstup
```
10 10 12
CCHCNHCCHN
NNCCCHHCCC
DNCCNNHHHC
NHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHH
HHHPCCCCCC
```

Výstup
```
Funkcia zachráň princezné vráti
NULL
Čas 0
```

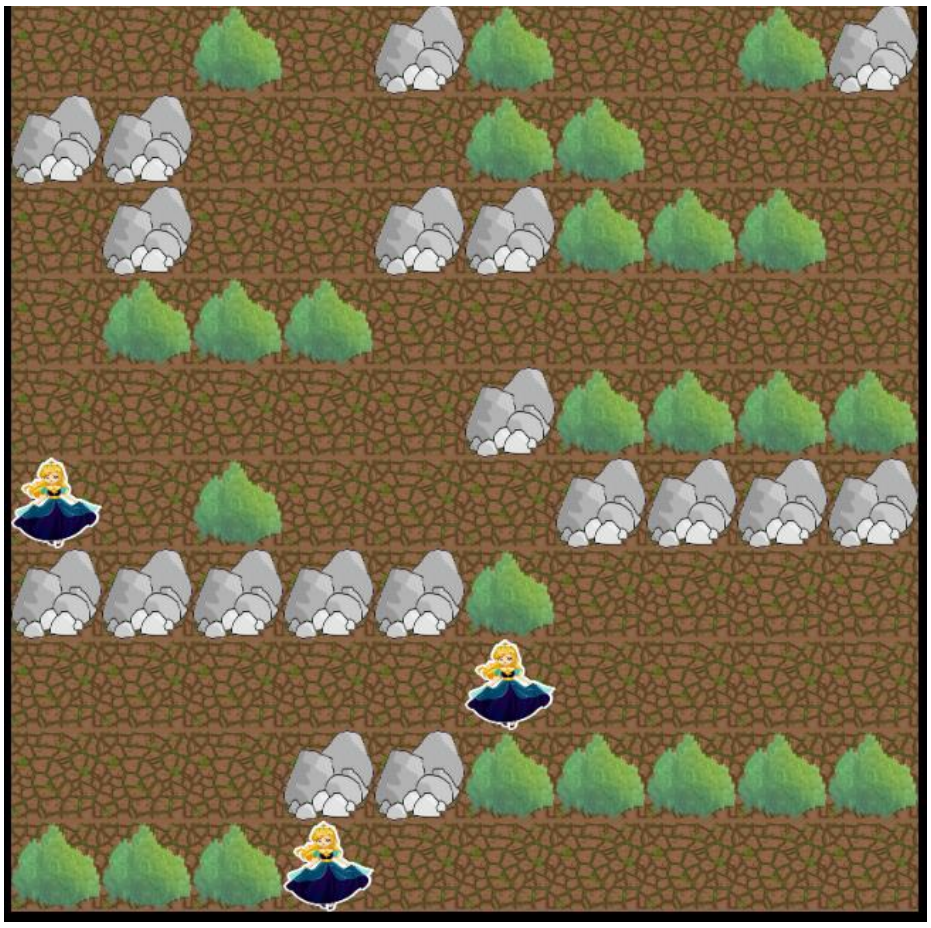


Test 3 drak nie je

Vstup

```
10 10 12
CCHCNHCCHN
NNCCCHHCCC
CNCCNNHHHC
CHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHH
HHHPCCCCCC
```

Výstup: 0



Test 4 začínam na skale

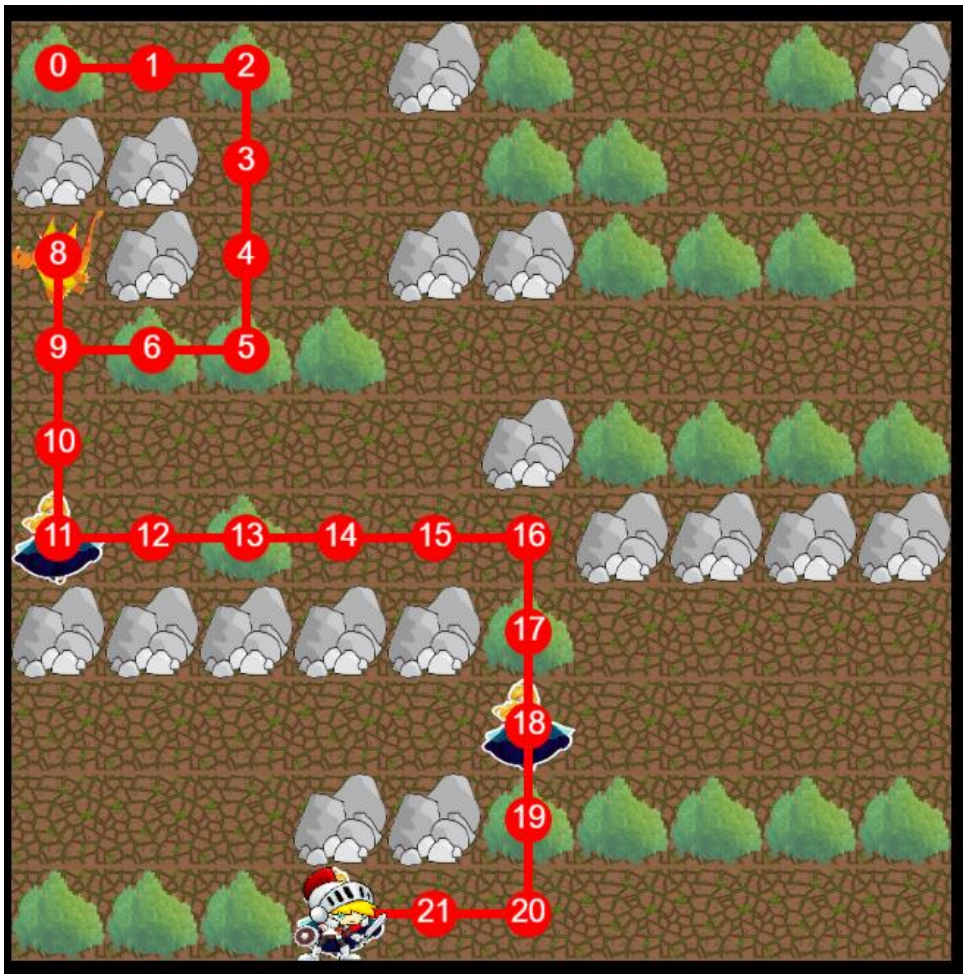
Vstup

```
10 10 12
NCHCNHCCHN
NNCCCHHCCC
DNCCNNHHHC
CHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHH
HHHPCCCCCC
```

Výstup: 0

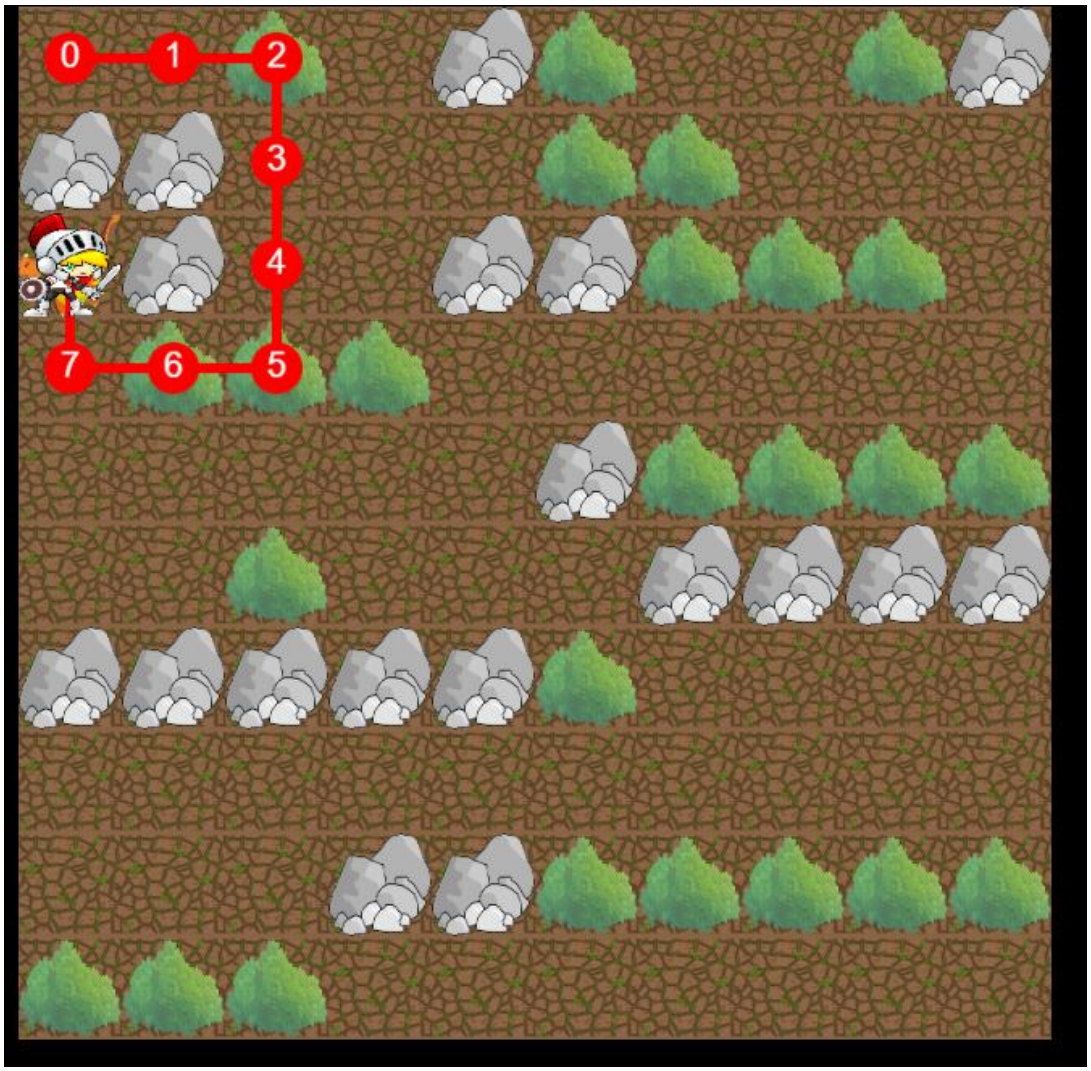
Test 5 začínam na húštine

Vstup	Výstup	0 5
10 10 12		1 5
HCHCNHCCHN	0 0	2 5
NNCCCHHCCC	1 0	3 5
DNCCNNHHHC	2 0	4 5
CHHHCCCCCC	2 1	5 5
CCCCCNHHHH	2 2	5 6
PCHCCCNNNN	2 3	5 7
NNNNNHCCCC	1 3	5 8
CCCCCPCCCC	0 3	5 9
CCCNNHHHHH	0 2	4 9
HHHPCCCCCC	0 3	3 9
	0 4	Čas 30



Test 6 mam draka nemám princeznú

Vstup	Výstup
10 10 12	0 0
CCHCNHCCHN	1 0
NNCCCHHCCC	2 0
DNCCNNHHHC	2 1
CHHHCCCCCC	2 2
CCCCCNHHHH	2 3
CCHCCCNNNN	1 3
NNNNNHCCCC	0 3
CCCCCCCCCC	0 2
CCCNNHHHHH	Čas: 12
HHHCCCCCCC	



Test 7 mam 5 princezien

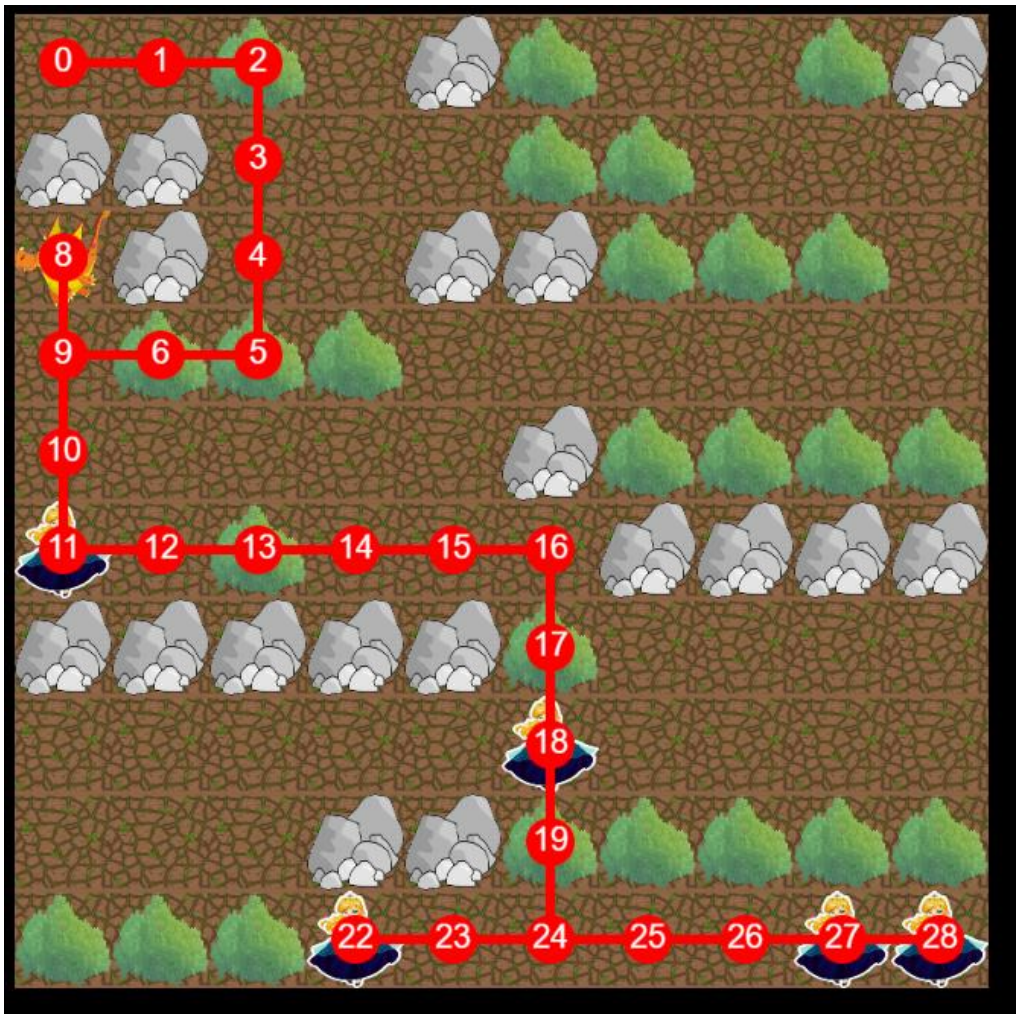
Vstup

```
10 10 12

CCHCNHCCHN
NNCCCHHCCC
DNCCNNHHHC
CHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHH
HHHPCCCCCC
```

Výstup:

```
0 0
1 0
2 0
2 1
2 2
2 3
1 3
0 3
0 2
0 3
0 4
0 5
1 5
2 5
3 5
4 5
5 5
5 6
5 7
5 8
5 9
4 9
3 9
4 9
5 9
6 9
7 9
8 9
9 9
Čas: 35
```



Test 8 mam draka mam 5 princezien a posledná je zablokovaná.

Vstup

Vystup:0

```
10 10 12
CCHCNHCCHN
NNCCCHHCCC
DNCCNNHHHC
CHHHCCCCCC
CCCCCNHHHH
PCHCCCNNNN
NNNNNHCCCC
CCCCCPCCCC
CCCNNHHHHN
HHHPCCPCNP
```

Test 9 Mega Test
Vstup a výstup je uložený v textovom dokumente pretože by to zabralo strašne veľa strán vo Worde.

Výsledok testovania:
Pri mojich testovacích scenároch som vyskúšal všetky možne situácie ktoré môžu nastáť no respektíve ktoré mňa napadli že môžu nastáť. Pri mojich malých vstupoch 10x10 sa to da pekne skontrolovať aj ručne bohužiaľ môj test 100x100 sa už ručne skontrolovať nedá a celkovo som neprišiel na nejaký spôsob ako to efektívne otestovať jedine čo ma napadlo je porovnať si môj výstup s výstupom môjho spolužiaka, cesty môžeme mat síce rôzne to záleží od toho ako prehľadával susedov ale cenu respektíve čas popolvára by sme mali mat rovnaký a ten nám obom v tomto mega teste vyšiel 273.