

# Popolvár

## Dátové štruktúry a algoritmy

### 1. Zadanie

Popolvár je jednoduchý chlapec z dediny, ktorý celé dni nerobí nič iné ako vynášanie popola zo sporáka v kuchyni a programovanie pre dobročinné účely. Dnes, ako tak surfoval po internete, sa k nemu dostala veľmi pohoršujúca správa. V neďalekej krajine uniesol drak viaceré princezné a schoval sa vysoko v horách, kde je veľmi ťažký prístup a chystá sa tam zajtra o polnoci všetky princezné zjesť.

Samozrejme, že sa tomu nemôžete len tak nečinne prizerať. Vďaka moderným technológiám máte k dispozícii aj mapu, ktorú rýchlo zverejnil kráľ – otec unesených nešťastníč. Vašou úlohou je teda prirodzene čo najskôr najprv zneškodniť draka a potom zachrániť všetky unesené princezné. Predpokladajte, že drak uniesol najviac 5 princezien.

Vašou úlohou je implementovať nasledovnú funkciu v programovacom jazyku C:

```
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 zachránil všetky unesené princezné a zneškodnil draka. Mapa je vysoká  $n$  políček a široká  $m$  políček ( $1 \leq n, m \leq 100$ ). Parameter  $t$  ( $0 \leq t \leq 106$ ) určuje, kedy od začiatku vášho hľadania sa drak zobudí a zje prvú princeznú. Keďže drak lieta veľmi rýchlo, spoľahnite sa, že ak sa vám ho nepodari zneškodniť dovtedy ako sa zobudí, princezné už nezachránite.

Prechod cez lesný chodník trvá jednu jednotku času a drak sa zobudí v  $t$ -tej jednotke času, kedy už bude neskoro. A nezabudnite, že najprv musíte zneškodniť draka, až potom môžete zachraňovať princezné (hoci by ste aj prechádzali predtým cez políčko kde je princezná). Veď ako by to bolo, keby ste bojovali s drakom pri zástupe princezien ako divákami...

Nájdenu cestu vráťte ako postupnosť súradníc (dvojíc celých čísel  $x, y$ , kde  $0 \leq x < m$  a  $0 \leq y < n$ ) všetkých navštívených políček. Na začiatku sa vždy nachádzate na políčku so súradnicami  $0,0$  a na poslednom navštívenom políčku musí byť jedna z unesených princezien. Ak existuje viacero rovnako dlho trvajúcich ciest, môžete vrátiť ľubovoľnú z nich. Výstupný argument `dlzka_cesty` nastavte na počet súradníc, ktoré ste vrátili ako návratovú hodnotu.

Implementujte vyššie uvedenú funkcionality čo možno najefektívnejšie. Pre hľadanie najkratšej cesty použite Dijkstrov algoritmus s binárnou haldou (Min Heap). Môžete použiť aj inú funkčnú metódu na nájdienie najkratšej cesty, príp. bez binárnej haldy, ale bude to za bodovú sankciu.

Príklad použitia vašej funkcie:

```
int i, *cesta = zachran_princezne(mapa, n, m, t, &dlzka_cesty);  
for (i=0; i<dlzka_cesty; ++i)  
    printf("%d %d\n", cesta[i*2], cesta[i*2+1]);
```

Riešenie zadania sa odovzdáva do miesta odovzdania v AIS do stanoveného termínu (oneskorené odovzdanie je prípustné len vo vážnych prípadoch, ako napr. choroba, o možnosti odovzdať zadanie oneskorene rozhodne cvičiaci, príp. aj o bodovej penalizácii). Odovzdáva sa jeden zip archív, ktorý obsahuje jeden zdrojový súbor s implementáciou a jeden súbor s dokumentáciou vo formáte pdf.

Dokumentácia musí obsahovať hlavičku (kto, aké zadanie odovzdáva), stručný opis použitého algoritmu s názornými nákresmi/obrázkami a krátkymi ukážkami zdrojového kódu, vyberajte len kód, na ktorý chcete extra upozorniť. Pri opise sa snažte dbať osobitý dôraz na zdôvodnenie správnosti vášho riešenia – teda dôvody prečo je dobré/správne, spôsob a vyhodnotenie testovania riešenia. Nakoniec musí technická dokumentácia obsahovať odhad výpočtovej (časovej) a priestorovej (pamäťovej) zložitosti vášho algoritmu.

Celkovo musí byť cvičiacemu jasné, že viete čo ste spravili, že viete odôvodniť, že to je správne riešenie, a viete aké je to efektívne.

Dôležité poznámky: Popolvár sa presúva len v štyroch smeroch (hore, dole, doľava, doprava). Ak nie je zadané inak, prechod cez políčko trvá štandardne jednu jednotku času. Teda cez políčka s drakom a

princeznou trvá prechod tiež jednu jednotku času. Do výsledného času sa započítavajú všetky políčka, cez ktoré Popolvár prejde. Ak teda začne na políčku (0,0), prejde cez políčko (1,0) a skončí na políčku (1,1), pričom všetky tri políčka obsahujú lesný chodník, tak Popolvárovi to trvá 3 jednotky času. Zneškodnenie draka je okamžitá akcia, ktorej trvanie je zanedbateľné vzhľadom na čas trvania prechodu cez políčko (teda ju zarátavame s nulovou dĺžkou trvania). Keďže Popolvár má celú radu ďalších princezien, ktoré musí ešte zachrániť v iných častiach sveta (na vstupe je viacero kráľovstiev, v ktorých chce zachrániť princeznú), musíte čo najskôr zachrániť princeznú (celkovo vrátane zneškodnenia draka), aby ste mohli čím skôr prejsť do ďalšieho kráľovstva. Na zadanej mape je vyznačených najviac 5 princezien.

Hodnotenie

Môžete získať 15 bodov, minimálna požiadavka 6 bodov.

12 bodov je za program a testovanie (za riešenie neobsahujúce Dijkstrov algoritmus s binárnou haldou max 9 bodov; z toho vlastné testovanie 2 body), 3 body sú za dokumentáciu (bez funkčnej implementácie 0 bodov), pričom body môžu byť výrazne ovplyvnené prezentáciou cvičiacemu (napr. keď neviete reagovať na otázky vzniká podozrenie, že to nie je vaša práca, a teda je hodnotená 0 bodov).

## 2. Algoritmus

### Dijkstrov algoritmus s binárnou haldou (Min Heap)

Hlavná myšlienka:

- 1) z počiatočnej bodky zapísať do min heap všetkých susedov do ktorých môžem ísť a označím ju ako navštívenú (Obrázok 2.1);



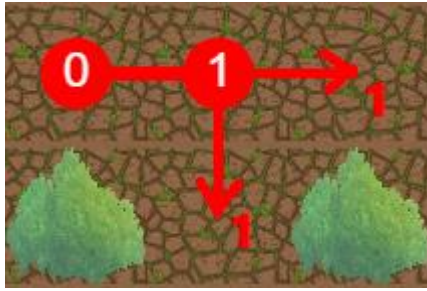
Obrázok 2.1 Susedia do min heap

- 2) vybrať si z min heap bodku s najmenšou veľkosťou;
- 3) prejsť do tej bodky (Obrázok 2.2);



Obrázok 2.2 Prechod do bodky

- 4) opakovať (Obrázok 2.3);



Obrázok 2.3 Opakovanie

Algoritmus dokončí kedy dôjde konečnej bodky.

Ak v min heape už je prázdny a algoritmus ešte nie v konečnej bodke, tak do nej nie je možné dôjsť.

### Implementácia

Aby chodiť mapou vytvorím novu mapu v ktorej bude uchovaný veľkosť, flag návštevy a predchádzajúce bod bodky.

```
typedef struct Node {  
    int weight;  
    int visited;  
    int prev_coords[2];  
}Node;
```

Pridávam do min heap susedne bodky, ak neboli navštívené.

```
for (int k = j - 1, p = i - 1; k <= j + 1; k += 2, p += 2) {  
    if ((k >= 0 && k < m) && (*(dependency_map + i * m + k))->weight >= 0 &&  
        (*(dependency_map + i * m + k))->visited == 0) {  
        (*(dependency_map + i * m + k))->visited = 1;  
        insertNode(createHeapNode(i, k, (*(dependency_map + i * m + j))->weight +  
            (*(dependency_map + i * m + k))->weight, i, j), minHeap);}  
    if ((p >= 0 && p < n) && (*(dependency_map + p * m + j))->weight >= 0 &&  
        (*(dependency_map + p * m + j))->visited == 0) {  
        (*(dependency_map + p * m + j))->visited = 1;  
        insertNode(createHeapNode(p, j, (*(dependency_map + i * m + j))->  
            weight + (*(dependency_map + p * m + j))->weight, i, j), minHeap);}  
}
```

Zoberám z min heap minimálnu bodku.

```
temp = extractMin(minHeap);
```

Pridávam veľkosť predchádzajúcej bodky a koordináty predchádzajúcej bodky.

```
*(dependency_map + temp->x * m + temp->y))->weight = temp->weight;  
*(dependency_map + temp->x * m + temp->y))->prev_coords[0] = temp->prev_x;  
*(dependency_map + temp->x * m + temp->y))->prev_coords[1] = temp->prev_y;
```

### Permutácie

Aby nájsť najkratšiu cestu do draka a všetkých princeznej potrebujem prejsť všetci permutácie princezien, napríklad, 4 princezné =  $4! = 24$  permutácie, 5 princeznej =  $5! = 120$  permutácií.

Na začiatku programu boli nájdený všetci hrdinovia a uchovaný do polia.

- 1) Vytvorím dvojrozmerné pole v ktorom zachovám permutácie.
- 2) V cykle hľadám cestu od draka do prvej princeznej v zozname permutácií.
- 3) Spojím cesty od popolvára do draka a od draka do princezné.
- 4) Hľadám cestu od spojenej cesty do ďalšej princeznej v zozname.
- 5) Spojím cesty a prejdem do 4.
- 6) Ak cesta ma menšiu veľkosť než predchádzajúca cesta uchovať cestu.

### 3. Testovanie


Aby otestovať program budem používať datasety rôznych rozmerov a tiež rôzne extrémne situácie.

#### 1. Mapa 10x10 z časom 12, 1 drak, 3 princezné.


```
10 10 12  
CCHCNHCCHN  
NNCCCHHCCC  
DNCCNNHHHC  
CHHHCCCCC  
CCCCNHHHH  
PCHCCNNNN  
NNNNNHCCCC  
CCCCPCCCC  
CCCNHHHHH  
HHHPCCCCC
```

```
cas vykonavania: 0.002 s
```

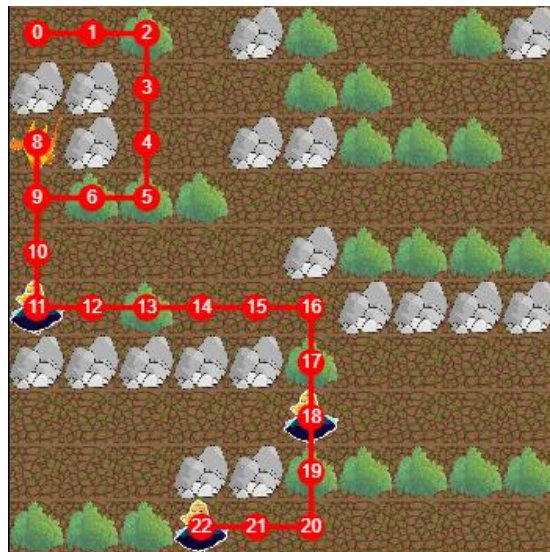
Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

po

>  popolvar.exe (2)

```
cas cesty: 29
```

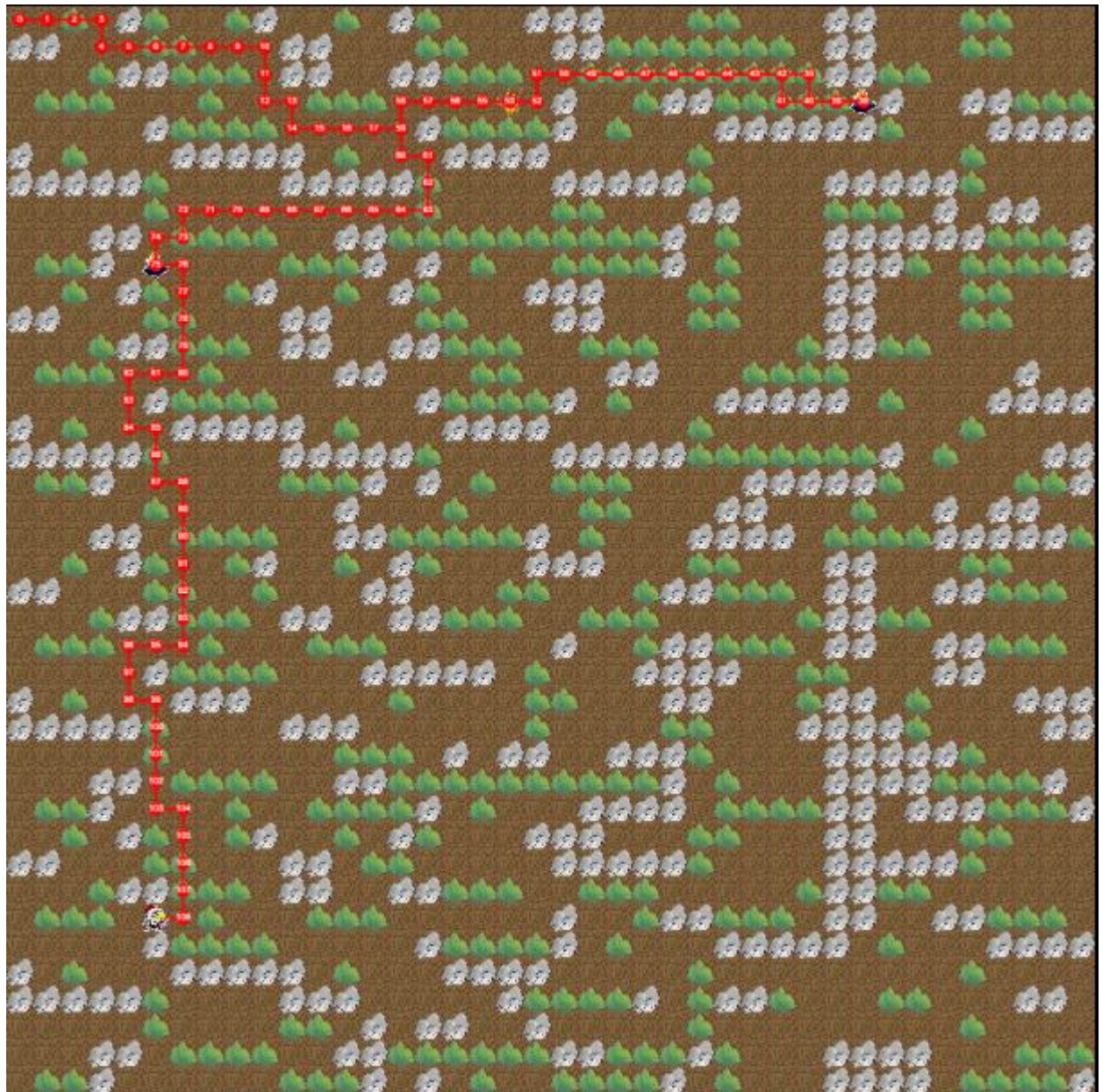


0% 6.7 MB

0% 7.1 MB




2. Mapa 40x40 z časom 100, 1 drak, 3 princezné




cas vykonavania: 0.017 s

Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

0%	6.7 MB
----	--------

po

>  popolvar.exe (2)

0%	11.0 MB
----	---------

cas cesty: 147




3. Mapa 20x80 z časom 100, 1 drak, 4 princezné




cas vykonavania: 0.085 s

Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

0% 6.7 MB

po

>  popolvar.exe (2)

0% 27.4 MB


cas cesty: 248

4. Mapa 20x80 z časom 100, 1 drak, 3 princezné.




cas vykonavania: 0.021 s

Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

0% 6.7 MB

po

>  popolvar.exe (2)

0% 11.1 MB

cas cesty: 220

5. Mapa 20x80 z časom 100, 1 drak, 1 princezná



cas vykonavania: 0.019 s




Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

0%	6.7 MB
----	--------

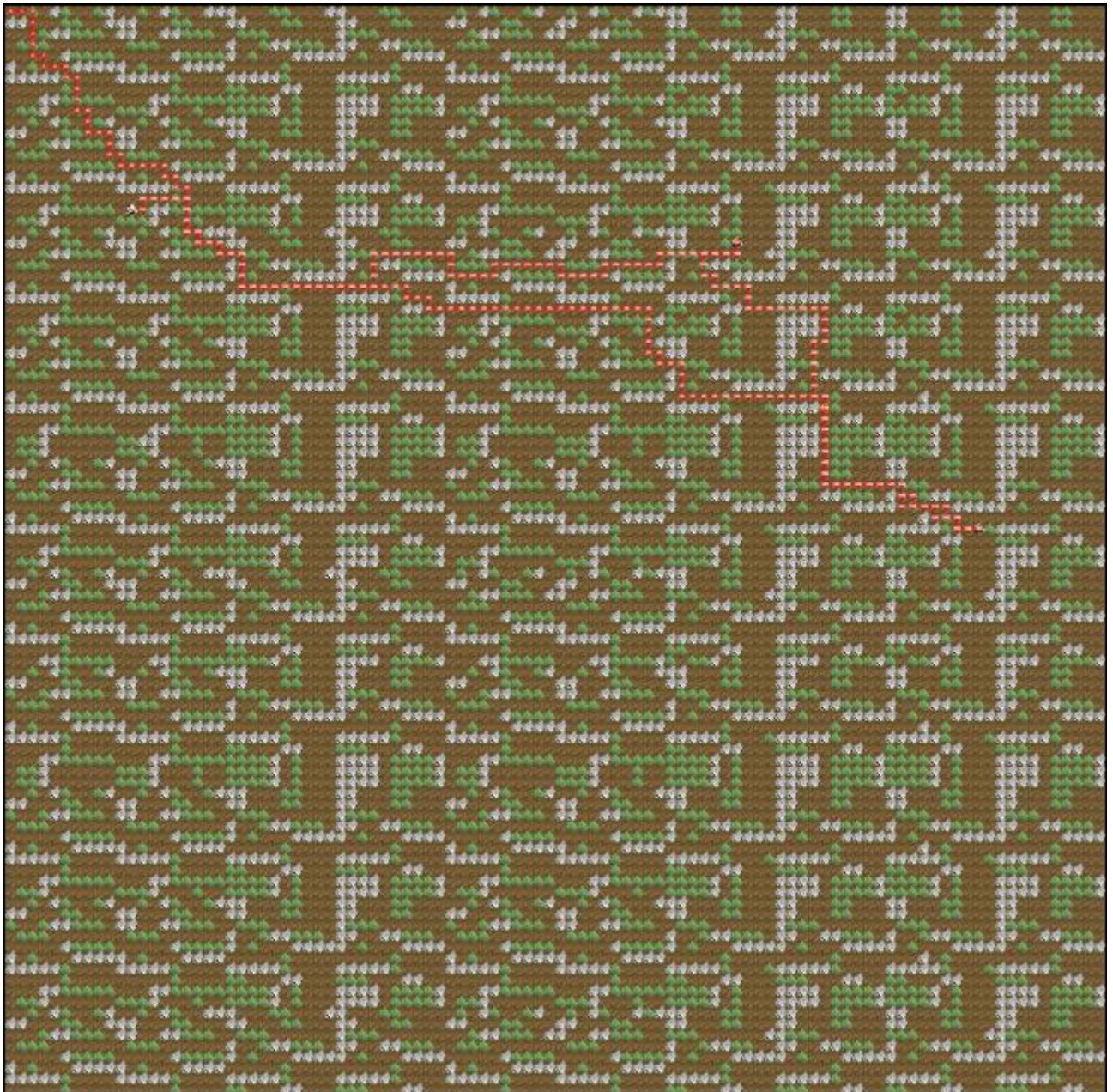
po

>  popolvar.exe (2)

0%	7.6 MB
----	--------


cas cesty: 150

6. Mapa 100x100 z časom 470, 1 drak, 3 princezné



cas vykonavania: 0.129 s


Obsadená pamäť:  
pred spustením algoritmu

>  popolvar.exe (2)

0.2%	6.7 MB
------	--------



po

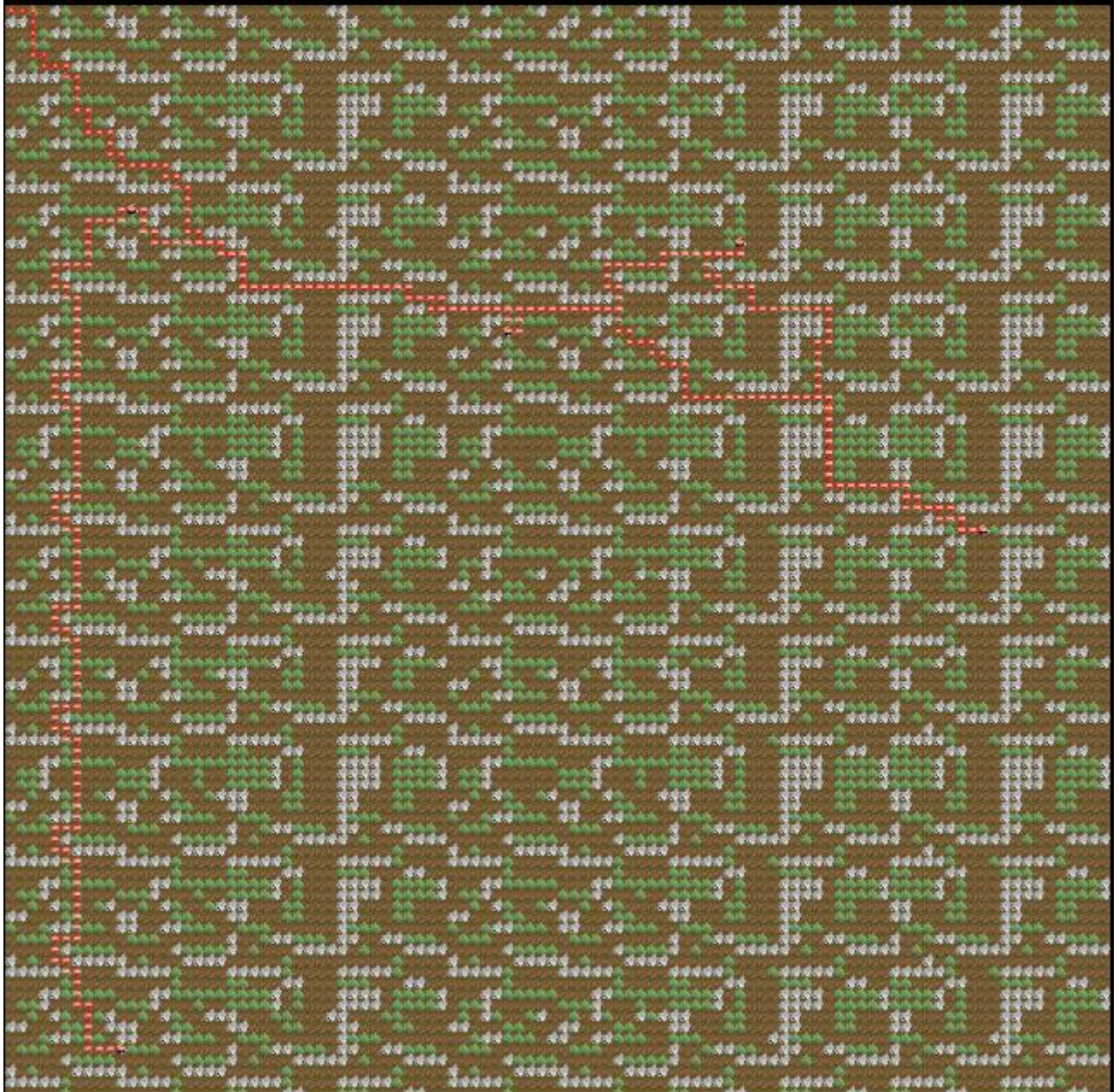
>  popolvar.exe (2)

0%

30.2 MB


cas cesty: 300

## 7. Mapa 100x100 z časom 470, 1 drak, 5 princeznej



cas vykonavania: 4.824 s

Obsadená pamäť:  
pred spustením algoritmu


>  popolvar.exe (2)

0%

6.7 MB

po



>  popolvar.exe (2)

0%

421.4 MB

cas cesty: 451

8. Mapa 10x10 z časom 12, 0 drakov, 3 princezné

Vypíše: Neexistuje žiaden drak!

9. Mapa 10x10 z časom 12, 1 drak, 0 princezné.

Vypíše: Neexistuje žiadna princezna!

10. Mapa 10x40 z časom 70, 1 drak, 2 princezné a 1 je zatvorená prekážkami.

Vypíše: Nemozem dosiahnuť hrdinu

11. Mapa 10x40 z časom 70, 1 drak, ktorý je zatvorený prekážkami, 3 princezné

Vypíše: Nemozem dosiahnuť hrdinu!

12. Mapa 20x80 z časom 100, 2 draka, 3 princezné

Vypíše: Viac ako 5 princeznej alebo viac ako 1 drak!

13. Mapa 20x80 z časom 100, 1 drak, 6 princeznej

Vypíše: Viac ako 5 princeznej alebo viac ako 1 drak!

14. Mapa 20x80 z časom 47(nestačí, aby zabil draka), 1 drak, 3 princezné

```
60 7
61 7
61 8
Nestihol si zabiť draka!
61 9
62 9
62 10
```

Vypíše:

4. Zhodnotenie

Z vyššie uvedených údajov môžeme usúdiť, že algoritmus nachádza draka a všetkých princeznej a vyberá najlacnejšiu cestu. Čas a pamäť závisia od veľkosti matice a počtu princezien. Čiže pri zväčšení matice a počtu princezien budú zväčšovať čas a pamäť vynaložené na tento proces.