Václav Luňák

Úkol 1.

1. Popis řešení

K řešení použijeme modifikovanou verzi algoritmu vlny. Ke každému stromu našeho lesa si budeme pamatovat hodnotu udávající, ve kterém čase začal hořet. Stromy, které se mají zpracovávat, budeme ukládat na frontu, počínaje prvními hořícími stromy. V každém kroku algoritmu pak nejprve vezmeme první strom na frontě a podíváme se na všechny jeho sousedy. Pokud najdeme nějaký, který není skála a ještě nebyl zapálen, nastavíme takovému sousedu čas zapálení o jedna vyšší než je náš a zařadíme ho na konec fronty. V momentě, kdy se fronta vyprázdní, hoří všechny stromy, které můžou hořet, tj. buď hoří celý les, nebo nehoří pouze části izolované skalami od všech ohnisek. Dobu, za kterou se zapálí celý les, získáme jako čas zapálení posledního stromu na frontě před vyprázdněním.

2. Pseudokód

```
main()
{
  for each strom
    casZapaleni(strom) = MAXINT;

  for each ohniskoPozaru
  {
    casZapaleni(ohniskoPozaru) = 0;
    PushNaFrontu(ohniskoPozaru);
  }

  dobaHoreniLesa = HoreniLesa();
}

HoreniLesa()
{
   while(frontaNeprazdna)
   {
      zkoumanyStrom = PopZFronty;
      ZapalSousedy(zkoumanyStrom);
   }
   dobaHoreni = casZapaleni(zkoumanyStrom);
   return dobaHoreni;
```

```
}

ZapalSousedy(strom)
{
   for each soused (strom)
      if (!jeSkala(soused) && casZapaleni(soused) > casZapaleni(strom)+1)
      {
         casZapaleni(soused) = casZapaleni(strom) + 1;
         PushNaFrontu(soused)
      }
}
```

3. Důkaz správnosti

Nejprve ukažme, že se algoritmus nemůže zacyklit. To ověříme tak, že každý strom je navštíven maximálně jednou. Ve chvíli, kdy by měl být navštíven podruhé, určitě už hoří. Protože stromy ukládáme a vybíráme z fronty, musí mít tento strom stejný nebo nižší čas zapálení než strom, který by ho navštívil. V algoritmu však navštěvujeme pouze stromy s vyšším časem zapálení, nemůže tato situace nastat.

Zároveň takto navštívíme všechny navštivitelné stromy, protože stromy dosud nenavštívené mají nastavenou nejvyšší možnou hodnotu, tudíš zapálený strom vždy navštíví všechny dosud nezapálené sousedy. Výslednou hodnotu získáme v momentě vyprázdnění fronty, jelikož v tu chvíli již žádný strom nemá souseda, kterého by mohl zapálit (při každém zapálení se strom zařadí na frontu).

Algoritmus vlny používáme, protože přesně odpovídá šíření ohně v lese. Použití například varianty DFS by bylo nevhodné, jelikož strom navštívený v DFS později by ve skutečnosti mohl začít hořet dříve, tudíž by se obtížněji určoval čas zapálení.

4. Časová a prostorová složitost

BFS algoritmus běží v čase $\theta(n + m)$. Tato variace ke každému stromu přidá pouze konstantní počet operací, tudíž se tato složitost nezmění. Navíc počet hran bude vždy menší nebo roven 4n (každý strom má nejvýše 4 sousedy), tedy výslednou časovou složitost můžeme odhadnout pomocí $\theta(n)$.
U každého stromu si pamatujeme pouze konstantní množství informací (pozice, jeStrom, čas zapálení). Prostorová složitost algoritmu je tedy taktéž $\theta(n)$.