Roboti v bludišti

1 Popis řešení

K nalezení cesty z bludiště (ne nutně nejkratší) použijeme prohledávání do hloubky. Budeme prohledávat stavový prostor, ve kterém vrcholy jsou dvojice pozic obou robotů a hrany jsou přechody mezi pozicemi. Přes hrany se budeme pohybovat pomocí příkazů S, J, V, Z.

Když se přesuneme do jiného stavu po stromové hraně, přidáme provedený příkaz na konec posloupnosti příkazů. Když se budeme z vrcholu vracet do rodiče, odebereme příkaz z posloupnosti. Jelikož se roboti mezi poli mohou hýbat v obou směrech, máme neorientované hrany, tedy neexistují dopředné ani příčné hrany. Pokud najdeme zpětnou hranu, budeme ji ignorovat, protože nám nezáleží na délce posloupnosti.

Prohledávání skončí, když se oba roboti dostanou do cílového vrcholu, nebo když projdeme všechny dosažitelné stavy.

2 Pseudokód

```
Seznam instrukce = new Seznam;
Seznam vysledek = new Seznam;
main(){
  if(PohniSe(startA, startB))
    writeln(vysledek);
  else
    writeln("Nelze");
}
bool PohniSe(a, b){
  if (navstiveno[a, b]) return false;
  if (a == cil && b == cil){
    vysledek = instrukce;
    return true;
  }
  navstiveno[a, b] = true;

  foreach smer in [S, J, Z, V]{
    if (a != cil && a+smer != stena)
       noveA = a+smer;
    else
       noveA = a;
```

```
    if (b != cil && b+smer != stena)
       noveB = b+smer;
    else
       noveB = b;

    instrukce.Pridej(smer)
    nalezeno = PohniSe(noveA, noveB);
    if (nalezeno)
       return true;
    instrukce.Odeber();
  }

  return false;
}
```

3 Důkaz správnosti

Když se prohledávání dostane do již navštíveného stavu, nemá smysl dále pokračovat, protože tyto stavy jsou identické. Algoritmus přidá instrukci do seznamu pouze, když se pohne daným směrem. Když se vyhledávání vrátí a nenajde cestu, zase ji ze seznamu odebere. Seznam tedy přesně odpovídá dosavadní prošlé cestě.

Když najdeme cestu do cíle, uložíme si celkovou cestu a všechna volání na zásobníku postupně hned skončí, tedy už dál prohledávání nepokračuje a ve výsledku máme nalezenou cestu do cíle. V opačném případě volání skončí poté, co projde všechny směry, tudíž jsme žádnou možnou cestu nevynechali.

4 Časová složitost

Má-li bludiště n polí, můžou roboti nabýt nejvýše n^2 různých stavů, tedy máme n^2 vrcholů. Jelikož existují jen 4 instrukce, má každý vrchol nejvýše 4 hrany, tedy $m \leq 4n^2$. Protože algoritmus je variantou DFS na takovémto grafu, jeho časová složitost je $O(n^2 + m) = O(n^2 + 4n^2) = O(n^2)$.

Pro snížení reálné časové složitosti si můžeme například uvědomit, že roboti jsou navzájem zaměnitelní, tedy jejich prohozením nijak nezměníme cestu do cíle. Toto změní počet stavů z n^2 na $\binom{n}{2}$. Asymptoticky se ovšem $O(\binom{n}{2}) = O(n^2)$.

5 Prostorová složitost

Stejnou argumentací jako u časové složitosti dojdeme u tohoto algoritmu k prostorové složitosti $O(n^2)$.