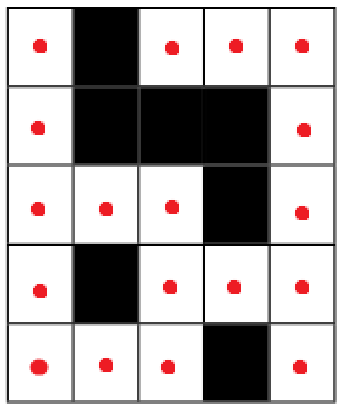
Navigace - Ondřej Švéda

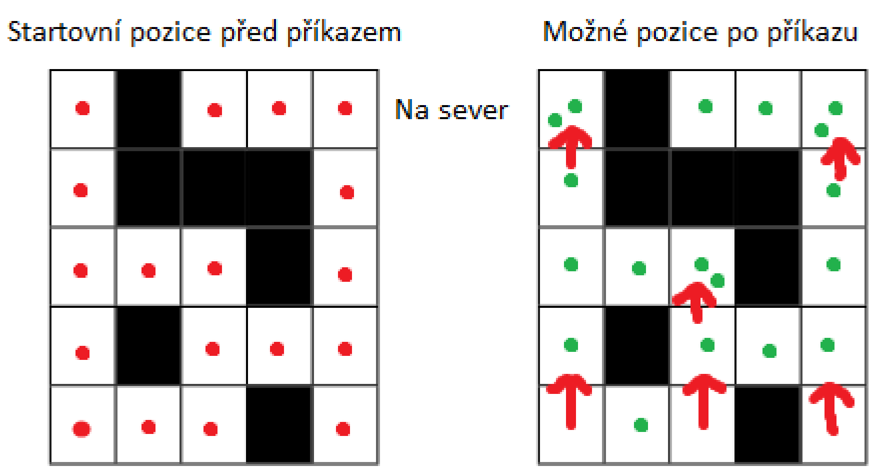
1: Pozorování o struktuře zadání

- Na začátku je důležité jasně vědět s jakou strukturou vlastně pracujeme.
- Major Wilford je někde ve městě -> neznámo kde
- Město je vlastně mřížka složená z volných políček a zdí, ve které se dá pohybovat z jednoho volného políčka na druhé buď nahoru, dolu, doleva nebo doprava
- Tím že nevíme pozici Wilforda tak není jasně definovaný začáteční bod ale jsou to vlastně všechny body. Znázorněno na mém velmi kvalitním obrázkovém popisu z malování:

Možné začáteční pozice Wilforda



- (tento obrázek platí pouze pro mapu města v zadání, pro jiné mapy bude rozmístění bodů jiné, avšak vždy muže začít na jakémkoli bílém poli)
- Ať začne Wilford kdekoliv ve městě, tak vždy se řídí našimi příkazy:
    - na sever, na jih, na západ a na východ
    - **ALE POZOR** když stojí vedle zdi a příkaz mu řekne ať jde směrem kde je zeď, Wilford se nepohne a zůstane na původní pozici
    - Viz (další super) obrázek



Některé body se sloučily

- Našim cílem je, abychom odnavigovali Wilforda na jeden konkrétní bod, který nám je známý -> ať začne kdekoliv, po sérii příkazů skončí v jednom bodě
- Jak bylo na předchozím obrázku vidět, když se bod zasekne o zeď a další bod do něj vstoupí, tak se "spojí" a pak už ať je zadána jakákoliv sekvence příkazů zůstanou pořád spolu **(VELMI DŮLEŽITÉ PRO ŘEŠENÍ)**

2: Přeformulované zadání

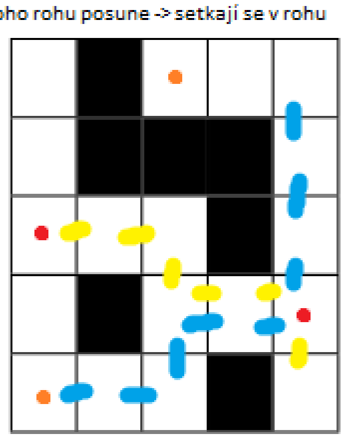
- Z původního složitého zadání navigace kde vlastně nevíme odkud ani kam se snažíme dostat se stalo postupné slučování množiny bodů:
  - Vstup:
    - Mapa bludiště, která obsahuje volná místa (bílý pole) N a zdi (černý pole)
    - Množina možných startů (bílých polí) P
    - Na začátku platí že P == N
  - Povolené pohyby:
    - Na sever -> posune všechny prvky množiny P nahoru(pokud tam neni zeď)
    - Na jih -> posune všechny prvky množiny P dolu(pokud tam neni zeď)
    - Na východ -> posune všechny prvky množiny P doprava(pokud tam neni zeď)
    - Na západ -> posune všechny prvky množiny P doleva(pokud tam neni zeď)
  - Cíl:
    - Najít sekvenci pohybu, takovou, aby na jejím konci počet prvků P = 1
    - Pokud to nejde vypsat "ajajaj"

3: Řešení -> Greedy pair merging algorithm s heuristikou

- Greedy znamená že bere při každém kroku aktuální nejlepší body, bez ohledu na to jak to posune s ostatními
- Algoritmus vždy pokud je to možné sloučí dvojici bodů (většinou i více protože se hýbou všechny body a aspoň na začátku je velká šance, že se bodů sloučí vice najednou/na více místech) , tím si zmenší velikost P a toto opakuje dokud není velikost P = 1 (všechny počáteční body jsou na jednom místě, tím pádem dokážeme s jistotou určit pozici Wilforda
- Inicializace:
  - Algoritmus dostane množinu všech možných počátečních pozic P
- Cyklus řešení:
  - While velikost P > 1:
    - 1. Výběr páru na sloučení a sekvence
      - Místo náhodné volby, která by sice fungovala, avšak mohla by být pomalejší, zvolíme dvojici bodů a sekvenci "chytře"

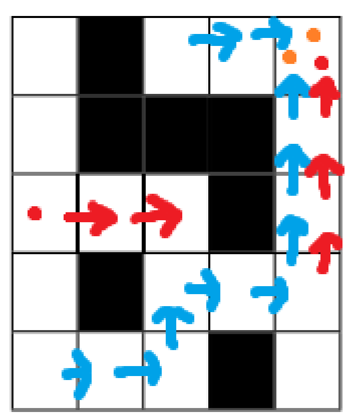
        - Použijeme multi-source BFS
            - Vyvtoříme produktový grafu (graf stavů (a’, b’), kde hrana = jeden příkaz na oba body najednou)
            - Do fronty BFS vložíme všechny možné dvojice bodů z množiny P jako počáteční vrcholy
            - Jeden společný BFS najde současně nejkratší sekvence pro sloučení pro každý pár
        - Pokud existuje dvojice bodů mezi kterou BFS nenajde cestu, tak algoritmus nikdy nebude schopen přesně určit pozici Wilforda, tím pádem se vypíše “ajajaj”
        - Pro každou nalezenou sekvenci S provedeme simulaci jejího použití na množinu P a zjistíme velikost množiny P’
            - Vybere se sekvence, která nejvíc sníží počet prvků v množině P (s danou dvojicí jde více prvků, které se k ní taky “připojí”)
            - Pokud je více sekvencí, které docílí stejného snížení velikosti množiny P, vybere se ta, která obsahuje méně kroků
    - 2. Aplikování vybrané sekvence
        - Zvolenou sekvenci přidáme do celkové sekvence, použité na množinu P aby se “sloučily” všechny body
        - Aplikujeme ji na všechny prvky aktuální množiny P, tím se zaručeně sníží její velikost minimálně o 1 (sloučení původní dvojice) ale většinou i o další prvky které se “přidají po cestě”
    - 3. Opakování
        - Tento cyklus se opakuje dokud velikost množiny P = 1
        - Příklad průběhu (další masterpiece z malování):

Modrá cesta končí v rohu, protože oranžový bod se při provedení předchozích příkazů do toho rohu posune -> setkají se v rohu



Žlutá cesta končí až v rohu, protože při provedení předchozích kroků se pravý červený bod posune dolu, tím pádem se mohou setkat až v rohu

Algoritmus vybere modrou cestu, přestože je delší, protože při jejím provedení se spojí oranžové body + pravý červený bod



- Až algoritmus doběhne -> když je velikost množiny P = 1 známe přesnou pozici Wilforda, tím pádem ho můžeme navigovat kamkoli chceme pomocí BFS

4: Důkaz korektnosti

- Konečnost:
  - Algoritmus určitě skončí, protože při každé iteraci se velikost P zmenší minimálně o 1 (pokud je na začátku iterace velikost P > 1)
  - Protože na začátku je velikost P = velikost N a každým krokem se množina P zmenšuje, maximální počet iterací je N - 1
- Korektnost:
  - Slučování bodů nikdy nerozšiřuje množinu P, jen ji zmenšuje (pokud to graf umožňuje a není v něm bílý bod obklopený ze všech 4 stran černými)
  - Pokud je nějaký bílý bod obklopen černými ze všech 4 stran -> BFS nenajde cestu a tím pádem algoritmus vyhodnotí, že úloha není řešitelná

5: Analýza časové složitosti

- Maximální počet iterací je N – 1 protože:
  - Na začátku P = N
  - Každá iterace zmenší P minimálně o 1 (pokud je úkol řešitelný)
  - O(N)
- Uvnitř každé iterace:
  - Maximální počet párů pozic (na začátku) -> N^2
  - Jeden společný BFS pro všechny páry v produktovém grafu -> O(N^2)
  - Simulace nejlepší sekvence O(N)
  - Celkově v 1 iteraci: O(N) + O(N^2) = O(N^2)
- Celkově v 1 iteraci O(N^2)
- Celkově:
  - Max počet iterací * časová komplexita 1 iterace
  - O(N) * O(N^2) = O(N^3)
- Sice je to O(N^3), takže u velkých měst se Wilford asi celkem projde, ale pořád to je polynomiální