Heuristika

Heuristika

- Heuristika (z řečtiny heuriskó, εύρίσκω – nalézt, objevit)
- zkusmé řešení problémů, pro něž neznáme algoritmus nebo přesnější metodu
- Heuristické řešení je často jen přibližné, založené na poučeném odhadu, intuici, zkušenosti nebo prostě na zdravém rozumu
- První odhad se může postupně zlepšovat, i když heuristika nikdy nezaručuje nejlepší řešení
- Zato je univerzálně použitelná, jednoduchá a rychlá

Heuristika

- Nejjednodušší heuristická metoda je pokus a omyl, kterou lze použít kdekoli, od připevnění šroubů na kolo až po řešení algebraických problémů.
- Několik běžně užívaných heuristických postupů:
  - Podívejte se na problém.
  - Pokud mu nerozumíte, zkuste si nakreslit obrázek.
  - Pokud nemůžete najít řešení, zkuste předpokládat, že ho máte a podívejte se, jestli z něj nemůžete získat postup („práce odzadu“).
  - Jestliže je problém abstraktní, zkuste nejdříve řešit konkrétní příklad.
  - Zkuste nejprve řešení obecnějšího problému ("paradox vynálezce": čím ambicióznější plán, tím lepší jsou vyhlídky na jeho dokončení)

Heuristika

- Máme nějaký problém, ke kterému chceme nalézt řešení
  - Obvykle prohledáváním grafu

Heuristika

- Máme nějaký problém, ke kterému chceme nalézt řešení
  - Obvykle prohledáváním grafu
- V momentě, kdy máme na výběr několik možných pokračování, které vyberem?

Heuristika

- Máme nějaký problém, ke kterému chceme nalézt řešení
  - Obvykle prohledáváním grafu
- V momentě, kdy máme na výběr několik možných pokračování, které vyberem?
  - Pokud umíme nějak určit, které pokračování je lepší, vybereme to nejlepší

Heuristika

- Máme nějaký problém, ke kterému chceme nalézt řešení
  - Obvykle prohledáváním grafu
- V momentě, kdy máme na výběr několik možných pokračování, které vyberem?
  - Pokud umíme nějak určit, které pokračování je lepší, vybereme to nejlepší
  - Tipneme si, třeba to vyjde = použijeme heuristiku

Heuristika

- Máme nějaký problém, ke kterému chceme nalézt řešení
  - Obvykle prohledáváním grafu
- V momentě, kdy máme na výběr několik možných pokračování, které vybereme?
  - Pokud umíme nějak určit, které pokračování je lepší, vybereme to nejlepší
  - Tipneme si, třeba to vyjde = použijeme heuristiku
  - Náhodně vybereme některé pokračování

Heuristika

- Odhadneme, kde je větší šance na nalezení řešení (nebo si alespoň myslíme, že by mohla být větší)
  - Použije se na určení pořadí, v němž se budou procházet varianty možných pokračování.

Heuristika

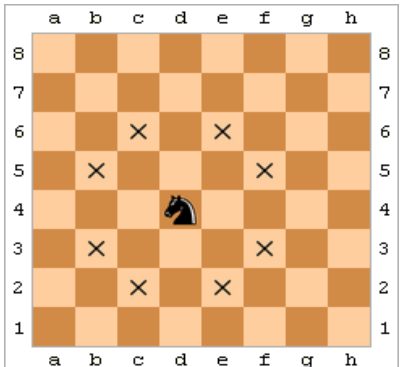
- Odhadneme, kde je větší šance na nalezení řešení (nebo si alespoň myslíme, že by mohla být větší)
  - Použije se na určení pořadí, v němž se budou procházet varianty možných pokračování.
- Hledáme-li jedno libovolné řešení, zvyšujeme dobrou heuristikou pravděpodobnost, že ho najdeme brzy
  - Pořadí listů ve stromu řešení se změní tak, že listy představující úspěšné řešení budou dříve prozkoumány (nakupí se více vlevo)
  - Projdeme třeba jen malou část stromu řešení, získáme časovou úsporu

Heuristika

- Odhadneme, kde je větší šance na nalezení řešení (nebo si alespoň myslíme, že by mohla být větší)
  - Použije se na určení pořadí, v němž se budou procházet varianty možných pokračování.
- Hledáme-li jedno libovolné řešení, zvyšujeme dobrou heuristikou pravděpodobnost, že ho najdeme brzy
  - Pořadí listů ve stromu řešení se změní tak, že listy představující úspěšné řešení budou dříve prozkoumány (nakupí se více vlevo)
  - Projdeme třeba jen malou část stromu řešení, získáme časovou úsporu
- Heuristika nic nezaručuje, použijeme ji na základě intuitivního odhadu programátora
  - Není-li heuristika dobrá, výpočet nezrychlí, ale o možnost nalézt řešení jejím použitím nepřijdeme

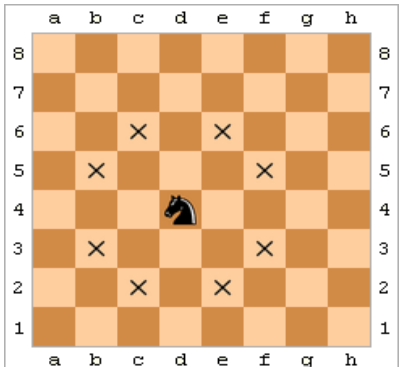
Proskákání šachovnice koněm

- Je dána výchozí pozice šachového koně, proskákat s ním postupně všechna pole na šachovnici, žádné přitom nenavštívit dvakrát



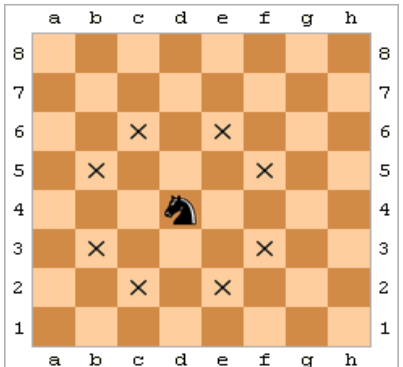
Proskákání šachovnice koněm

- Je dána výchozí pozice šachového koně, proskákat s ním postupně všechna pole na šachovnici, žádné přitom nenavštívit dvakrát
  - v každé pozici zkoušíme postupně provést koněm tah na všechna dosud nenavštívená sousední pole



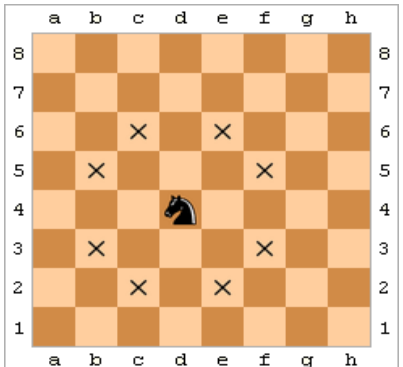
Proskákání šachovnice koněm

- Je dána výchozí pozice šachového koně, proskákat s ním postupně všechna pole na šachovnici, žádné přitom nenavštívit dvakrát
  - v každé pozici zkoušíme postupně provést koněm tah na všechna dosud nenavštívená sousední pole
  - pro každý tah potom budeme (rekurzivně) hledat pokračování řešení z nové pozice



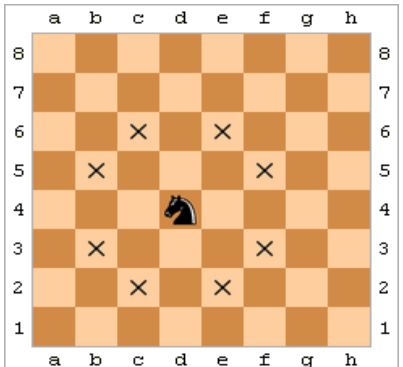
Proskákání šachovnice koněm

- Existuje účinná heuristika



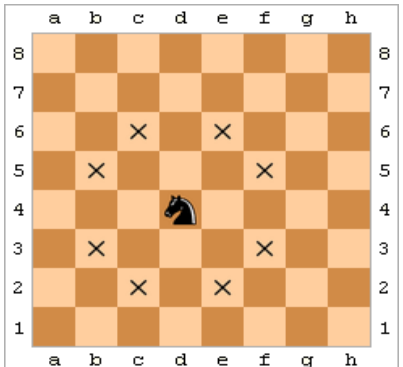
Proskákání šachovnice koněm

- V každé pozici provádět dříve vždy ty skoky, které vedou do pozic s menším počtem dalších pokračování
- Proč?



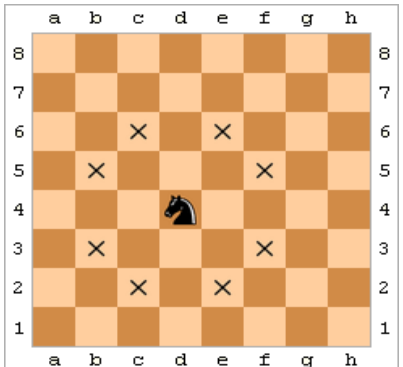
Proskákání šachovnice koněm

- V každé pozici provádět dříve vždy ty skoky, které vedou do pozic s menším počtem dalších pokračování
- Proč?
  - Protože si nezablokujeme možnost pokračování
  - Protože dříve vyřadíme možnosti, které stejně k cíli nepovedou

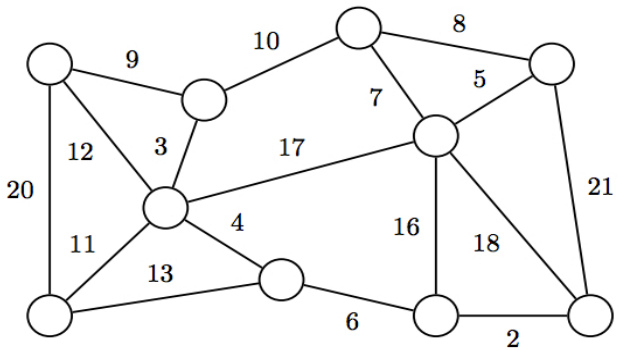


Proskákání šachovnice koněm

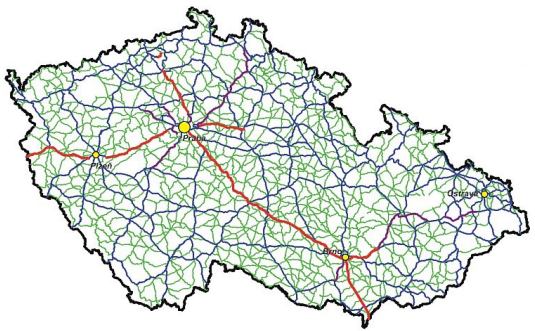
- V každé pozici provádět dříve vždy ty skoky, které vedou do pozic s menším počtem dalších pokračování
- Proč?
  - Protože si nezablokujeme možnost pokračování
  - Protože dříve vyřadíme možnosti, které stejně k cíli nepovedou
- Použití heuristiky způsobí zrychlení výpočtu o několik řádů



Hledání nejkratší cesty do cíle

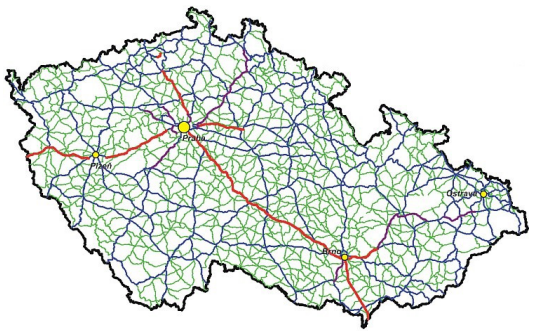


Hledání nejkratší cesty do cíle



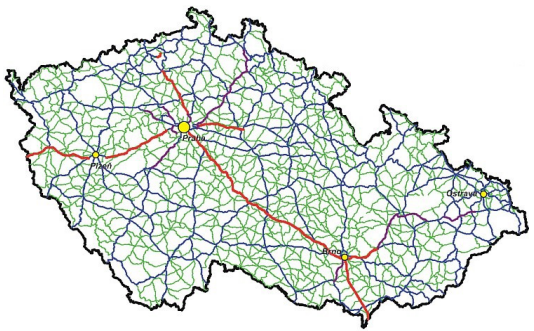
A* algoritmus

- Algoritmus pro vyhledávání optimálních cest v kladně ohodnocených grafech
- Funguje jako Dijkstrův algoritmus, přidává navíc heuristický prvek



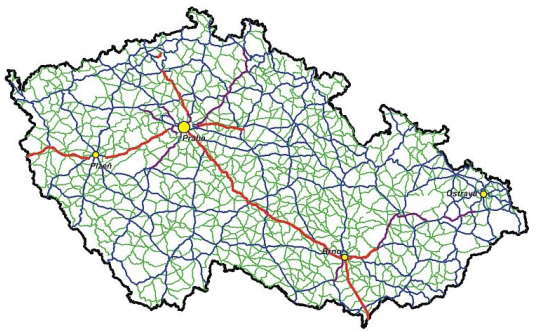
A* algoritmus

- Algoritmus pro vyhledávání optimálních cest v kladně ohodnocených grafech
- Funguje jako Dijkstrův algoritmus, přidává navíc heuristický prvek
  - Odhad vzdálenosti z aktuálního vrcholu do cíle



A* algoritmus

- A* používá hladový princip pro nalezení optimální cesty z počátečního vrcholu do koncového vrcholu
  - Optimální cestou se rozumí nejkratší, nejrychlejší, nejlevnější atd. cesta
- Používá funkci obvykle označenou f(x)= g(x) + h(x), která ohodnocuje vrcholy
  - Funkce g(x) představuje vzdálenost mezi počátečním vrcholem a vrcholem x
  - Funkce h(x) představuje heuristickou funkci – odhad vzdálenosti z vrcholu x do koncového vrcholu
  - Heuristika h(x) musí být přípustná, tzn. nesmí nadhodnocovat vzdálenost k cíli
    - V navigaci může být použita jako heuristika vzdálenost vzdušnou čarou, jelikož je to fyzicky nejkratší možná cesta



Vlastnosti heuristiky

- Přípustná, tzn. nesmí nadhodnocovat (vzdálenost k cíli)

  heuristika h splňuje podmínku

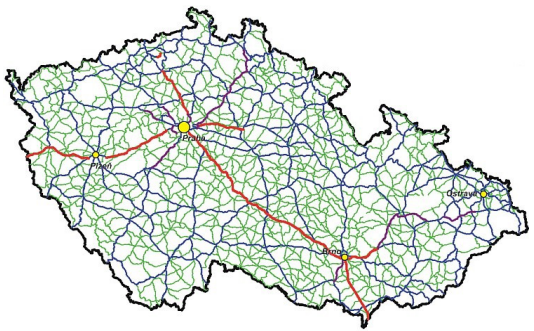
  $$h(x) < v(x)$$

  kde v(x) délka nejkratší cesty z vrcholu x do cílového vrcholu
- Monotónní (též označována jako konzistentní)

  heuristika h splňuje podmínku

  $$h(x) < d(x,y) + h(y)$$

  pro každou hranu x, y grafu (d je délka této hrany)
  - V případě monotónní heuristiky algoritmus A* navštíví každý vrchol maximálně jednou



A* algoritmus

- Samotný algoritmus pak probíhá následovně. Je vytvořena a udržována prioritní fronta otevřených, tj. ještě nenavštívených uzlů. Čím menší je hodnota f(x) pro daný uzel x, tím vyšší má prioritu.
- V každém kroku algoritmu je uzel s nejvyšší prioritou odebrán z prioritní fronty a jsou spočítány hodnoty f(y) a h(y) pro jeho sousední uzly y. Tyto uzly jsou pak přidány do prioritní fronty anebo jsou sníženy jejich hodnoty, pokud ve frontě už jsou a nové hodnoty jsou nižší.
- Algoritmus pokračuje, dokud nemá koncový uzel c menší hodnotu f(c), než libovolný jiný uzel z fronty, nebo dokud není tato fronta prázdná.
- Hodnota f(c) koncového uzlu je poté délkou nejkratší cesty grafem.
- Pokud je potřeba znát i konkrétní cestu, je nutné udržovat si i seznam uzlů na této cestě. Pro udržování stačí pamatovat si v každém uzlu jeho (libovolného) předchůdce na nejkratší cestě.