IV130 Přínosy a rizika inteligentních systémů

Řešení problémů vyhledávání

22. března 2024

Aktéři řešící problémy

- **Aktér řešící problémy** je aktér, který **hledá** *posloupnost akcí* vedoucích k žádoucímu **cíli**
- **Stavy** prostředí mohou být atomické, vnímané pouze prostřednictvím senzorů – postup k cíli se nazývá *řešením problémů*
- Stavy se strukturou vedou na tzv. **plánování** (*plánující aktéři*), např. dokazování teorémů

Vyhledávání

- Řešení problémů jako hledání cesty z **počátečního stavu** do jednoho nebo více **cílových stavů** ve **stavovém prostoru**
- K dispozici jsou **akce**, které aktér vykonává
- **Přechodový model** popisuje účinky akcí, dvojice (*stav*,*akce*) se zobrazuje do nového *stavu*
- S akcemi je zpravidla spojen **náklad** (cena) akce (přes **nákladovou funkci**)
- **Cesta** je posloupnost akcí, **řešení** je cesta z počátečního stavu do cílového
- Optimální řešení je řešení s minimálním nákladem

Algoritmy vyhledávání v aktérovi

- Algoritmy vyhledávání typicky předpokládají prostředí:
  - Epizodické
  - Plně pozorovatelné
  - Statické
  - Diskrétní
  - Známé

- Variantami algoritmu jsou
  - **Informovaný** (aktér odhaduje vzdálenost k cíli)
  - **neinformovaný** (bez takového odhadu)

- Cíl je **určitý**
- Hledáme optimální řešení
- Reprezentace stavového prostoru je konstruována až při řešení, v minimálním rozsahu

Standardizované typy problémů

- **Standardizované typy problémů** dovolují uvažovat jistou typizaci prostředí s ohledem na používané přístupy/algoritmy
- Vhodné pro ilustraci používaných přístupů

- **Realistické problémy** odpovídají skutečně používaným systémům, zpravidla se od sebe výrazně liší (např. závislost na senzorech, akcích, typu zařízení, atd.)

Standardizovaný typ problémů: svět založený na mřížce

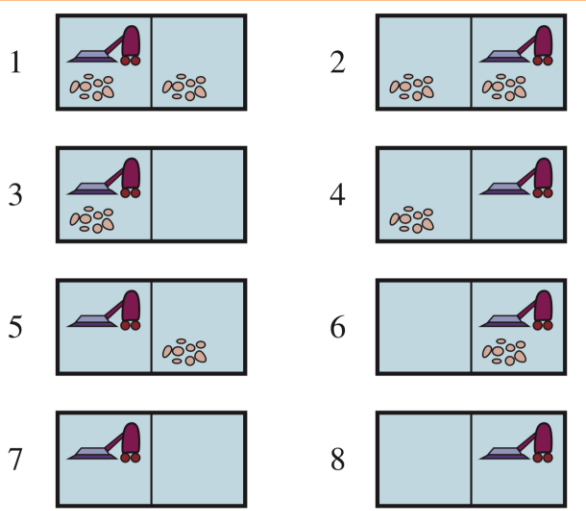
- **Mřížka** je dvojrozměrné pravoúhlé pole čtvercových **polí**, mezi nimiž se aktér pohybuje
- Může obsahovat struktury typu překážek,
- Pohyb aktéra může být horizontální, vertikální, diagonální, atd.

- Pole mohou obsahovat objekty, s nimiž aktér nějak manipuluje
- Překážka nebo stěna může aktérovi bránit v pohybu na toto místo
- Atd.

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Mřížka: svět vysavače

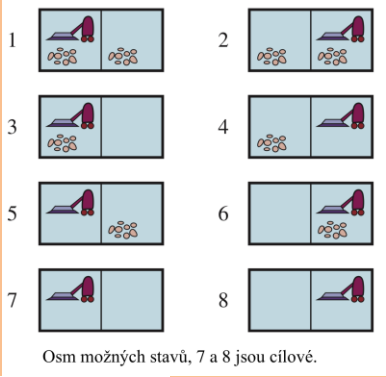
- **Stavy** popisují, jaké objekty jsou na jakých polích (aktér nebo smetí)
- *Polohou* aktéra je jedno ze dvou polí, sousední pole může obsahovat smetí nebo nikoli, tj. celkem 2x2x2 stavů
- **Počátečním stavem** je kterýkoli z nich
- **Akce** jsou: vysávej, postup doleva a postup doprava
- Přechodový model: sání S odstraní smetí, pohyby L, R změní polohu aktéra nebo jsou bez efektu
- **Cílový stav**: libovolný stav se všemi poli bez smetí
- Náklady akcí: každá akce má cenu 1



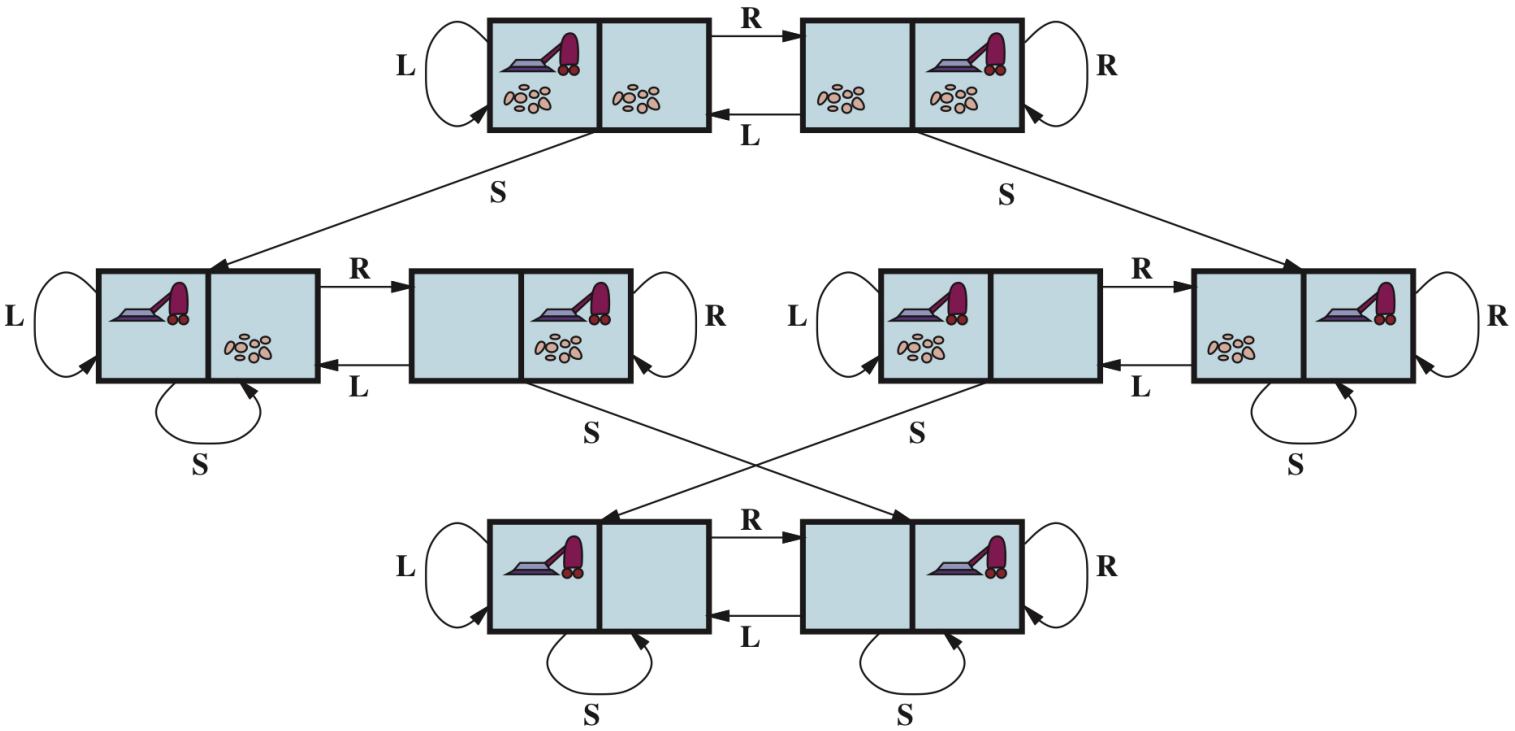
Osm možných stavů, 7 a 8 jsou cílové.

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Mřížka: svět vysavače



Osm možných stavů, 7 a 8 jsou cílové.

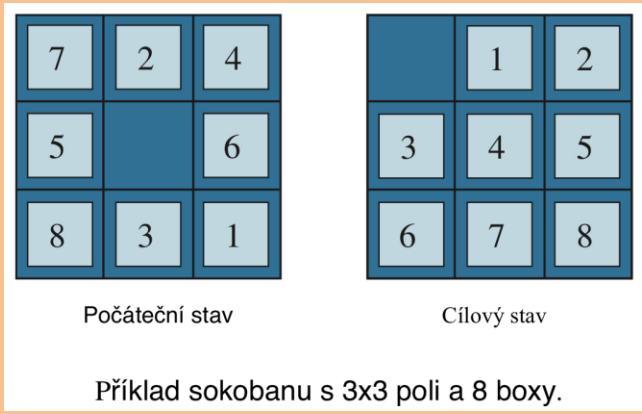


Stavový prostor pro svět vysavače se dvěma poli.
Celkem 8 stavů a tři akce pro každý stav:
L = *doleva*, R = *doprava*, S = *vysaj*.

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Mřížka: posunování boxů („sokoban“ = skladník)

- **Stavy:** specifikace polohy každého očíslovaného boxu
- **Počátečním stavem** je nějaký zvolený stav
- **Akce** jsou: posun zvoleného boxu doleva, doprava, nahoru nebo dolů, pokud to jde,
- **Přechodový model** je dán přechody mezi stavy v důsledku zvolené akce
- **Cílový stav**: nějaký zvolený stav (v modelu 3x3 nemusí být dosažitelný)
- Náklady akcí: každá akce má cenu 1



Příklad sokobanu s 3x3 poli a 8 boxy.

- Pro *n* políček a *b* boxů je zde *nxn*!/(*b*!(*n*-*b*)!) stavů (např. pro 8x8 a 15 boxů se jedná o více než 10^{16} stavů)

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Mřížka: přirozené číslo utvořené operacemi druhé odmocniny, faktoriálu, celé části a čísla 4

- Problém formulovaný Donaldem Knuthem v roce 1964 („Representing numbers using only 4“, Mathematics Magazine, 37, 308-310): vyjádřete libovolné přirozené číslo pomocí čísla 4 a libovolné posloupnosti operací druhé odmocniny, celé části a faktoriál
- **Stavy** jsou přirozená čísla
- **Počáteční stav**: 4
- **Akce**: druhá odmocnina, celá část, faktoriál z celých čísel
- **Přechodový model**: dán matematickými operacemi
- **Cílový stav**: zadané přirozené číslo
- **Náklady akcí**: 1

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}} \right\rfloor = 5$$

- Stavový problém je zde nekonečný, cesty mohou být extrémně dlouhé:
- Nejkratší cesta od 4 k 5 vede přes (4!)! = 620 448 401 733 239 439 360 000

Problémy z reálného světa: nejkratší cesta v mapě

- **Stavy**: Poloha v mapě
- **Počáteční stav**: Lokace uživatelova bydliště
- **Akce**: Přejezd z místa na místo (od křižovatky na křižovattku) po cestě vyznačené na mapě
- **Přechodový model**: Posun mezi lokacemi na mapě odpovídající přejezdu mezi dvěma křižovatkami po cestě na mapě
- **Cílový stav**: Místo, kam má uživatel dojet
- **Cena akce**: Doba vyžadovaná k přejezdu s ohledem na stav silnice a stupeň dopravy

Problémy z reálného světa: cesta letadlem

- **Stavy**: Poloha v mapě (zejména letiště) a aktuální čas, dále pak údaje o ceně letenky, status předchozího letu, atd. pro kalkulaci ceny návazné části cesty
- **Počáteční stav**: Domovské letiště uživatele
- **Akce**: Let z daného letiště v libovolné třídě, odlet po dané době, příjezd na letiště s dostatečným předstihem před odletem, atd.
- **Přechodový model**: Stav z využití letu bude mít cílové letiště jako novou polohu a přílet letu jako nový čas
- **Cílový stav**: Město, kam uživatel chce letět
- **Cena akce**: Kombinace ceny letenky, doby čekání, délky letu, doba pohybu po letišti, atd.

Problémy z reálného světa:

- **Cestovní itineráře**: úlohou může být postupně navštívit různé lokace a zorganizovat návaznou dopravu mezi nimi (zejména třeba problém obchodního cestujícího jako speciální případ)
- **Návrh podoby obvodu VLSI**: rozmístění milionů součástek a popojení do čipu s cílem minimalizovat rozměry, provozní náklady a ztráty
- **Navigace robota**: zobecnění problému hledání cesty/itineráře; zde je třeba počítat is možností chybné funkce senzorů, přenosu momentu z motoru na pohyb po podlaze nebo manipulace s objekty, atd.
- **Automatická práce** robota na výrobní lince: zvládání poloh součástek a manipulace s nimi, kontrola možnosti zadané akce fyzicky realizovat, atd.

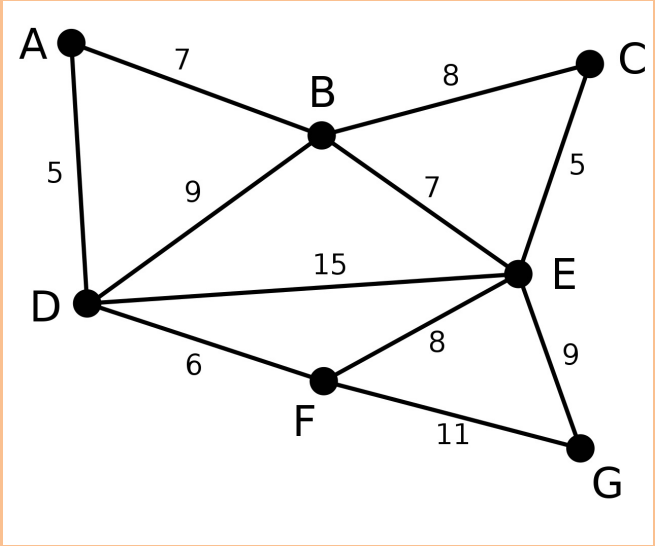
Prohledávací algoritmy

- Prohledávací algoritmy zpracují na vstupu vyhledávací problém a na výstupu vracejí řešení, nebo příznak selhání

- Běžné východisko je vytvoření **stromu vyhledávání** ke grafu stavového prostoru – uzly ve stromu vyhledávání odpovídají uzlům ve stavovém prostoru a hrany odpovídají akci provedené v daném stavu; kořen tohoto stromu odpovídá výchopímu stavu
- Strom vyhledávání zachycuje potenciálně nekonečné množství stavu světa a přechody mezi nimi
- Uzly stromů generování se zpravidla generují podle potřeby během výpočtu a v co nejmenším rozsahu

Grafy stavových prostorů

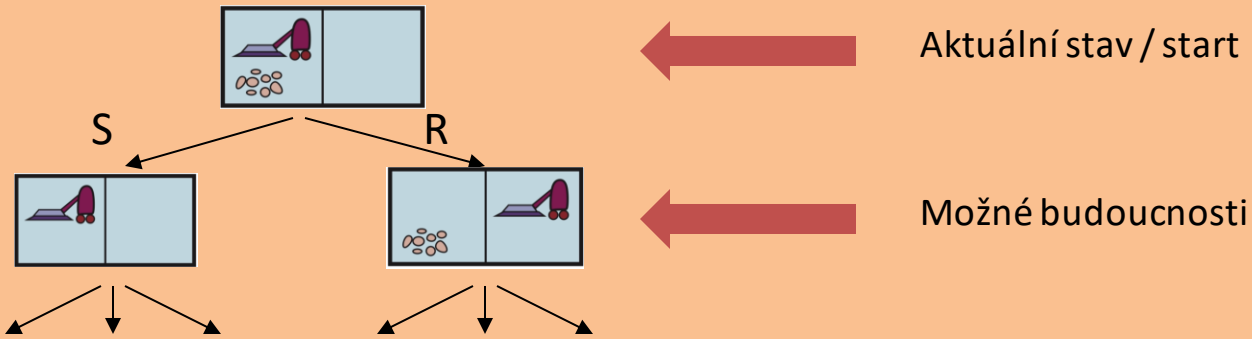
- Graf stavového prostoru je matematická reprezentace problému hledání
  - Uzly jsou (abstraktní) stavy světa
  - Hrany reprezentují následníky (výsledky akcí)
  - Cílový stav je množina uzlů (potenciálně jen jeden uzel)
  - Váhy hran odpovídají nákladům na akci s hranou spojené
- V grafu stavového prostoru se každý stav vyskytuje právě jednou
- Málokdy se takový graf konstruuje v paměti (je příliš velký), ale jako abstraktní idea je užitečný



Příklad malého stavového prostoru

(cesty mezi body s nákladem každé z hran připsaným k ní)

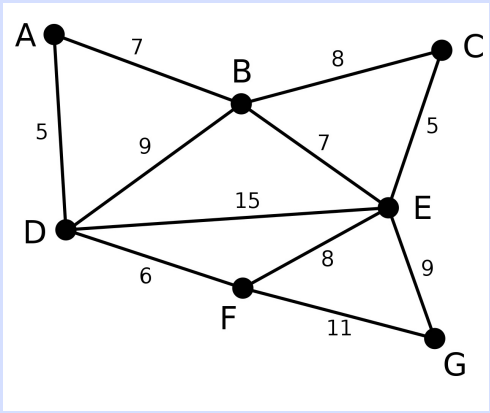
Stromy prohledávání



- Strom prohledávání:
  - Strom možných plánů a jejich výstupů pro různé posloupnosti akcí
  - Počáteční stav je uzel v kořenu
  - Bezprostřední potomci odpovídají následníkům
  - V uzlech jsou zapsány stavy, ale odpovídají plánům, jak se do těchto stavů dostat
  - Prakticky nikdy celý takový strom nestavíme

Graf stavového prostoru vs. strom prohledávání

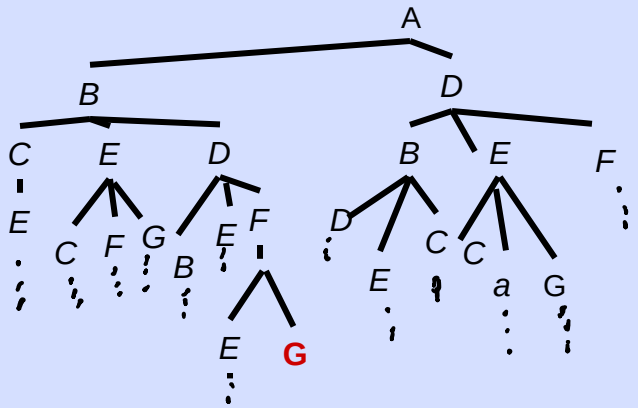
Graf stavového prostoru



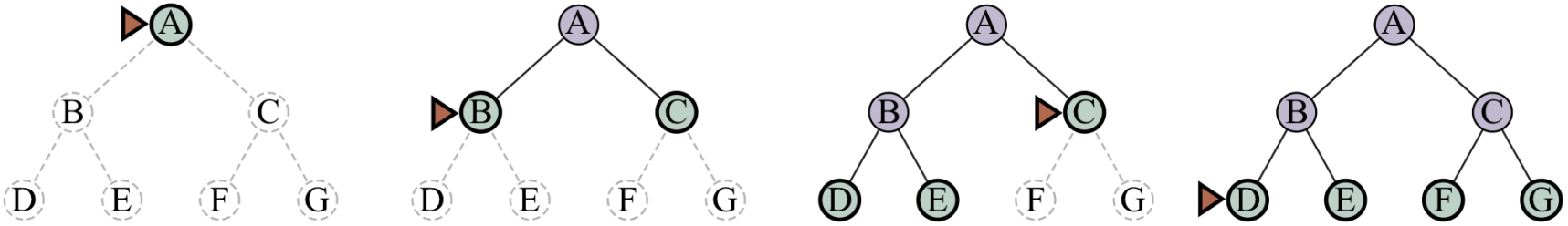
Každý uzel stromu prohledávání je celá cesta v grafu stavového prostoru

Obojí konstruujeme až podle potřeby – a tak málo rozlehlé, jak to jde.

Strom prohledávání



Neinformované prohledávací strategie: hledání do šířky

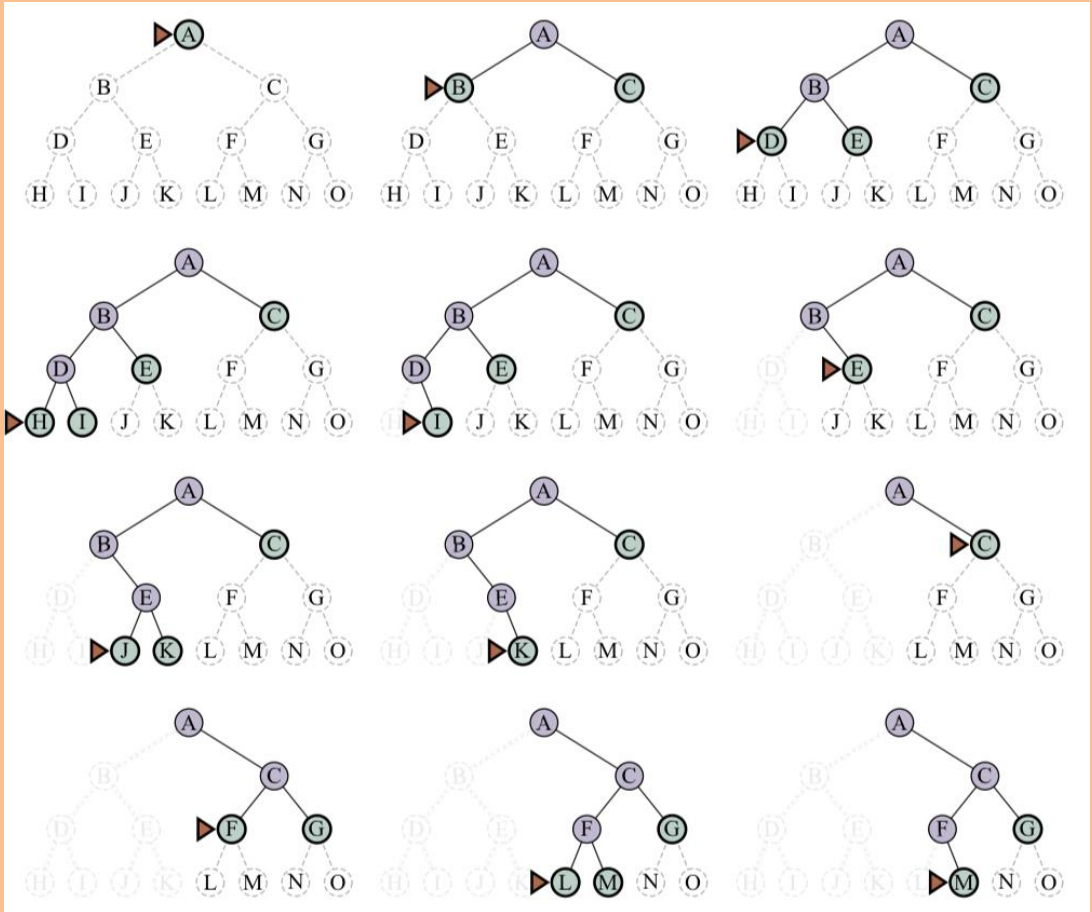


Hledání do šířky na jednoduchém binárním stromě. Uzel, který má být expandován, je označen trojúhelníčkem.

- Exponenciální požadavky na paměť
- Úplná metoda, ale použitelná jen na malé případy

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Neinformované prohledávací strategie: hledání do hloubky

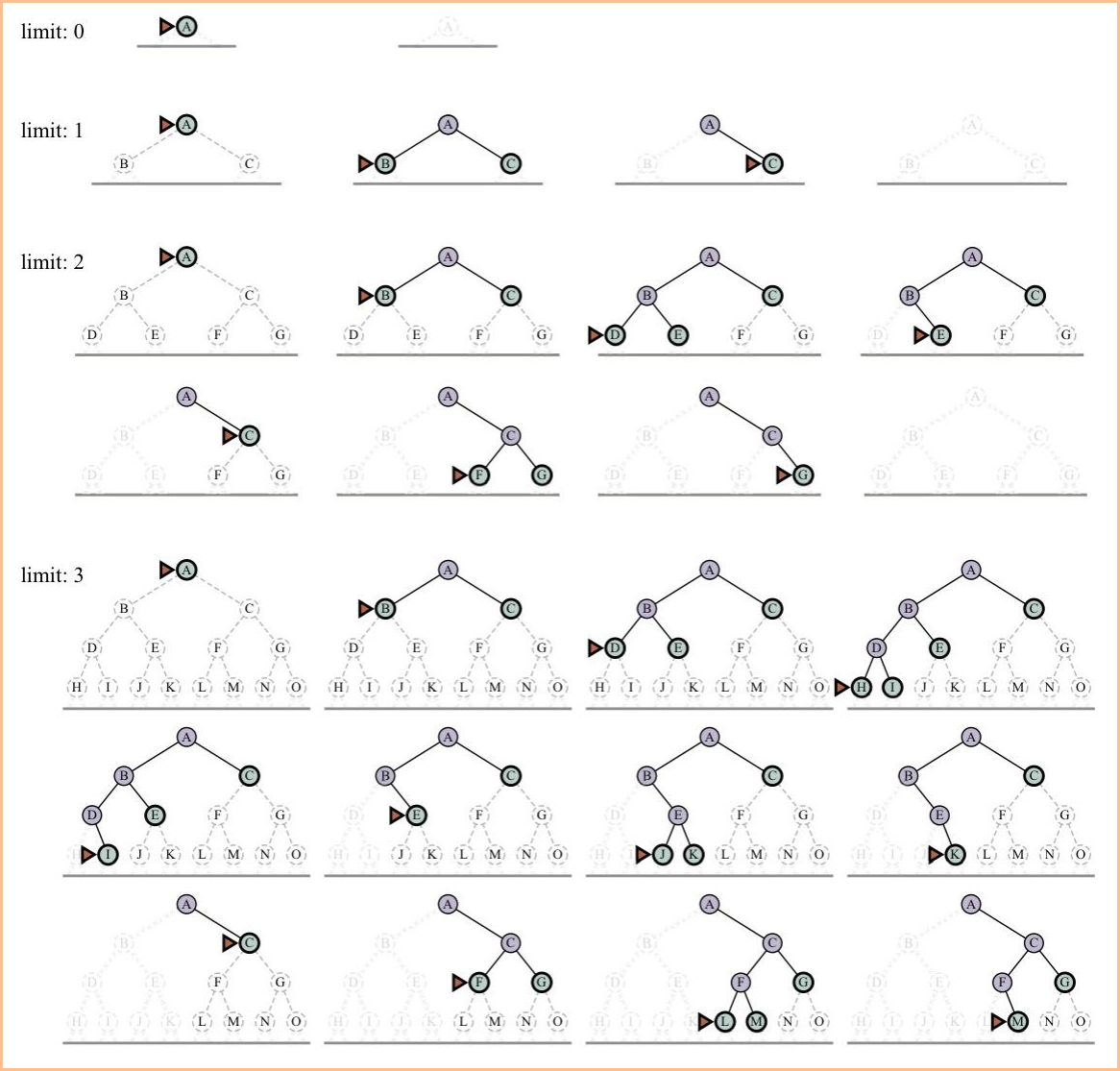


Několik kroků (zleva doprava, shora dolů) provádění hledání do hloubky během hledání do hloubky na binárním stromě z kořene A. Generované ještě neexpandované uzly jsou zelené, fialová barva označuje uzly již dříve expandované a čárkovaně jsou uzly, které přijdou na řadu v budoucnu. Trojúhelníček označuje expandovaný uzel.

- Hledání do hloubky není úplné
- Hledá první řešení vlevo ve struktuře stromu
- Při cyklických vazbách může uváznout na nekonečné větvi bez návštěvy zbytku stromu
- Efektivní z hlediska datových struktur
- Velmi efektivní v kombinaci s backtrackingem

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

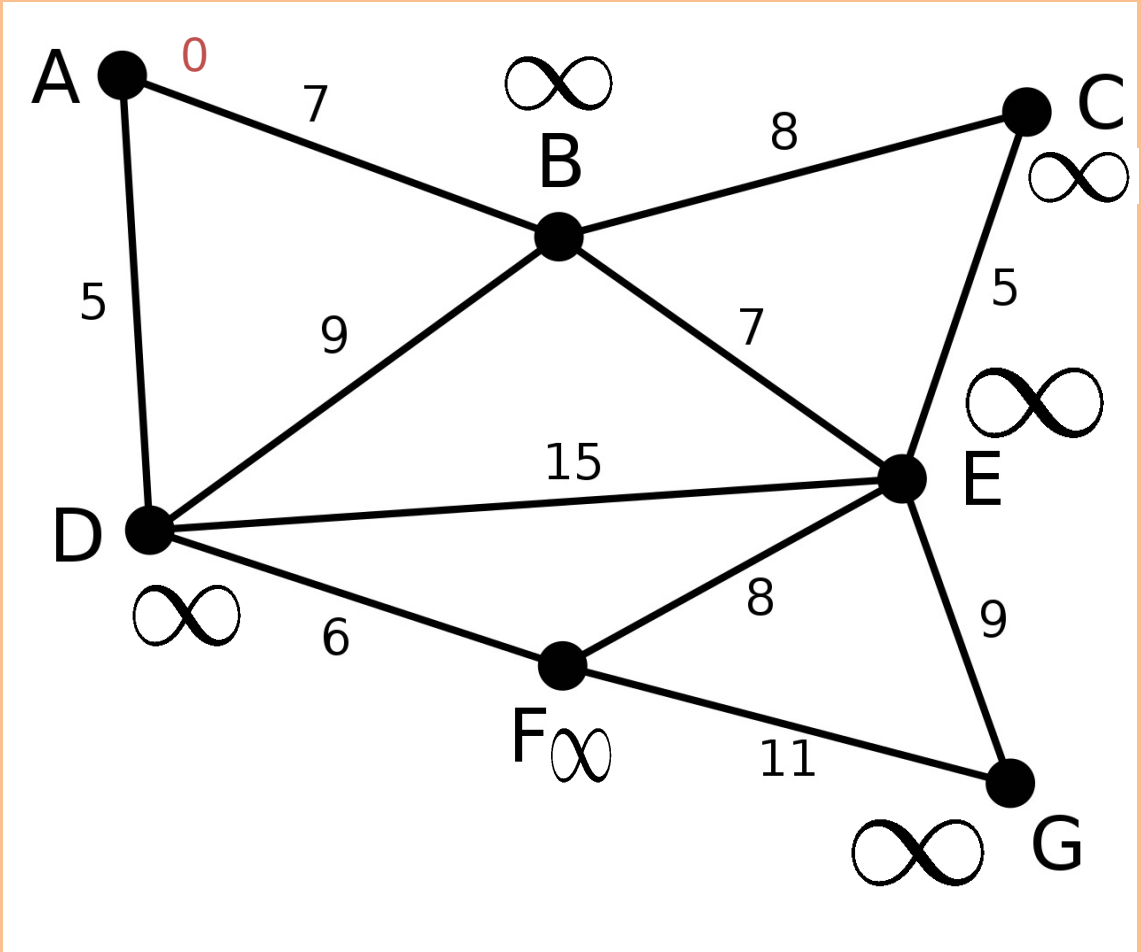
Neinformované prohledávací strategie: iterativní prohlubování



- Čtyři iterace iterativního prohlubování pro limity 0, 1, 2, 3. Zeleně jsou generované uzly, trojúhelníčkem je oznmačen expandovaný uzel.
- Úplné hledání za cenu opakování části expanzí z předchozích úrovní
- Nejvhodnější metoda tam, kde je strom prohledávání velký, nevejde se do paměti a není známa hloubka hledaného řešení.

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

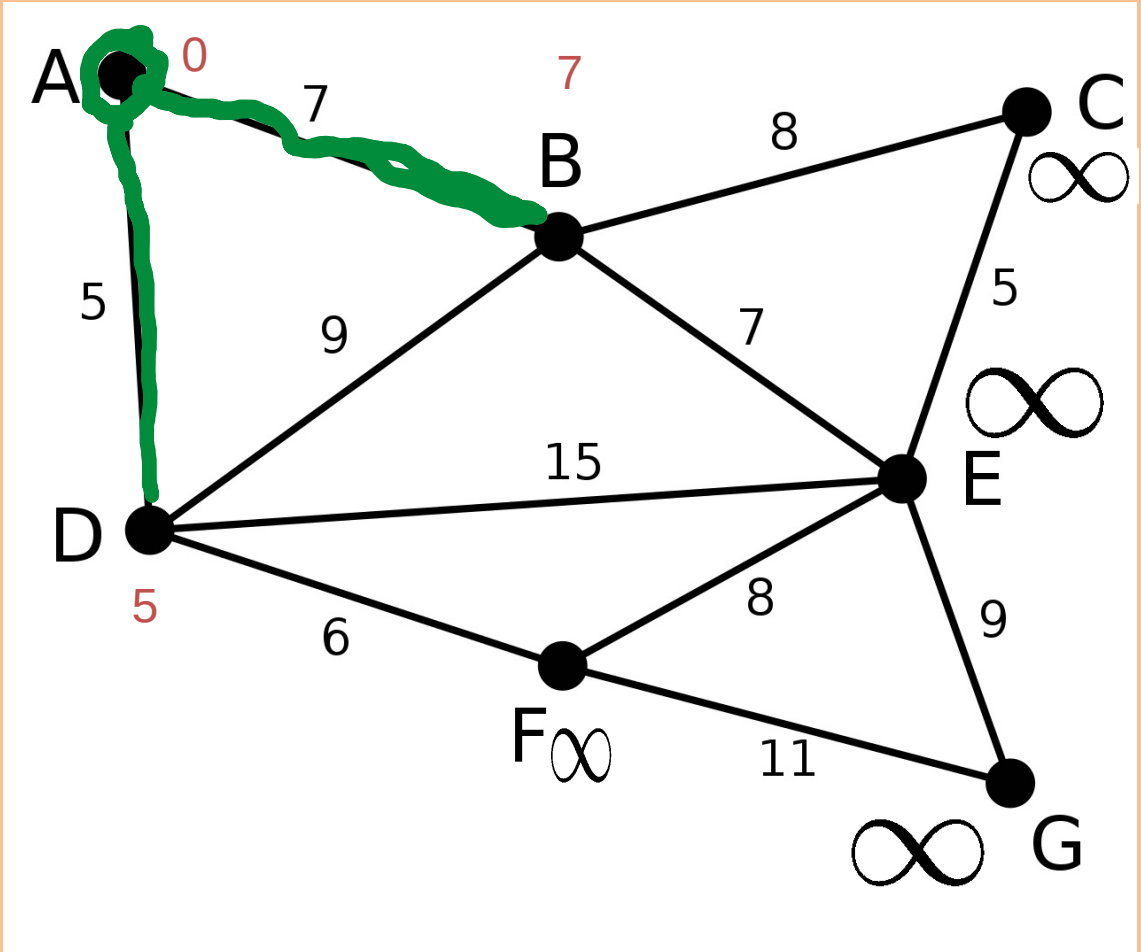
Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

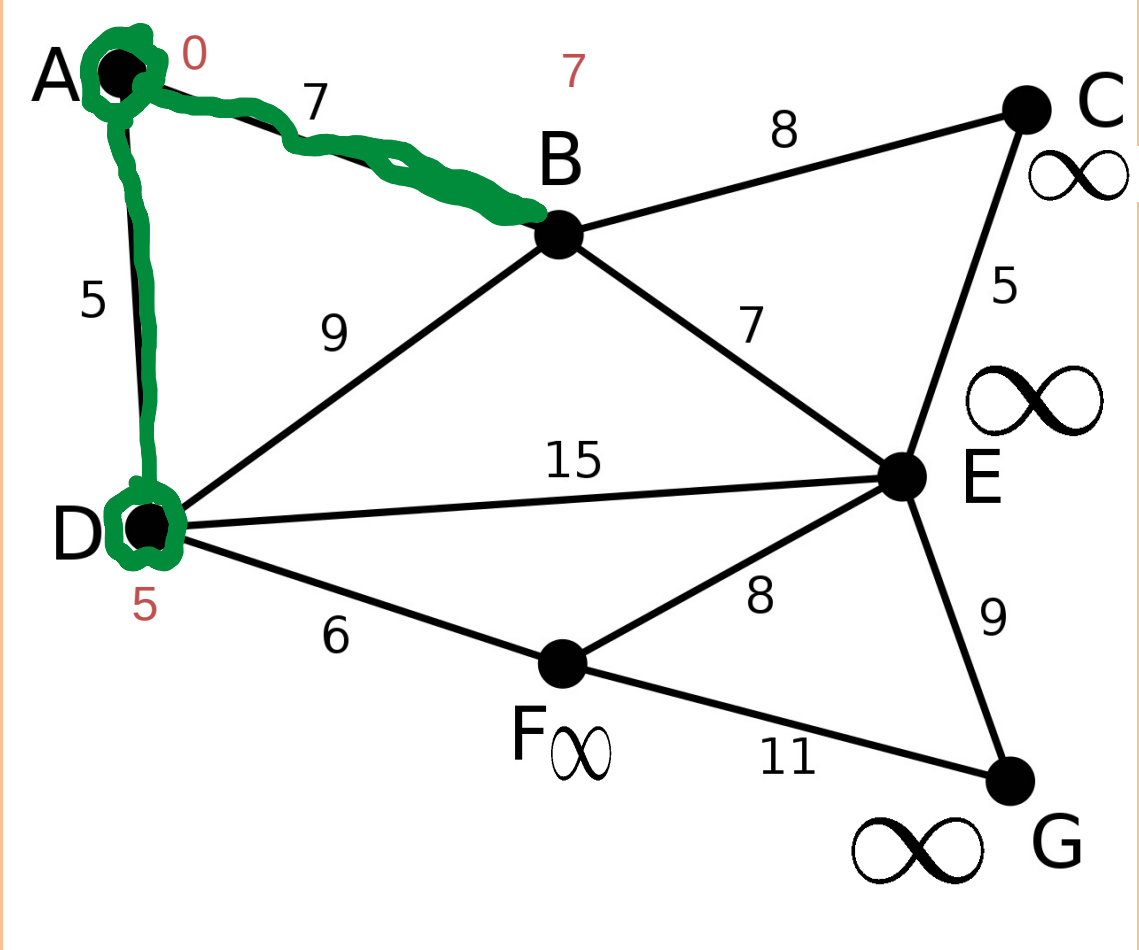
- Zvolte uzel přiléhající k nejlepšímu uzlu bez nekonečna a při jeho expanzi jeho bezprostředním následovníkům přiřaďte menší z čísel, které mají jako hodnotu a součtu hodnoty expandovaného uzlu s cenou hrany od něj.
- Pokračujte do expanze všech uzlů (bude značeno zeleně).

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



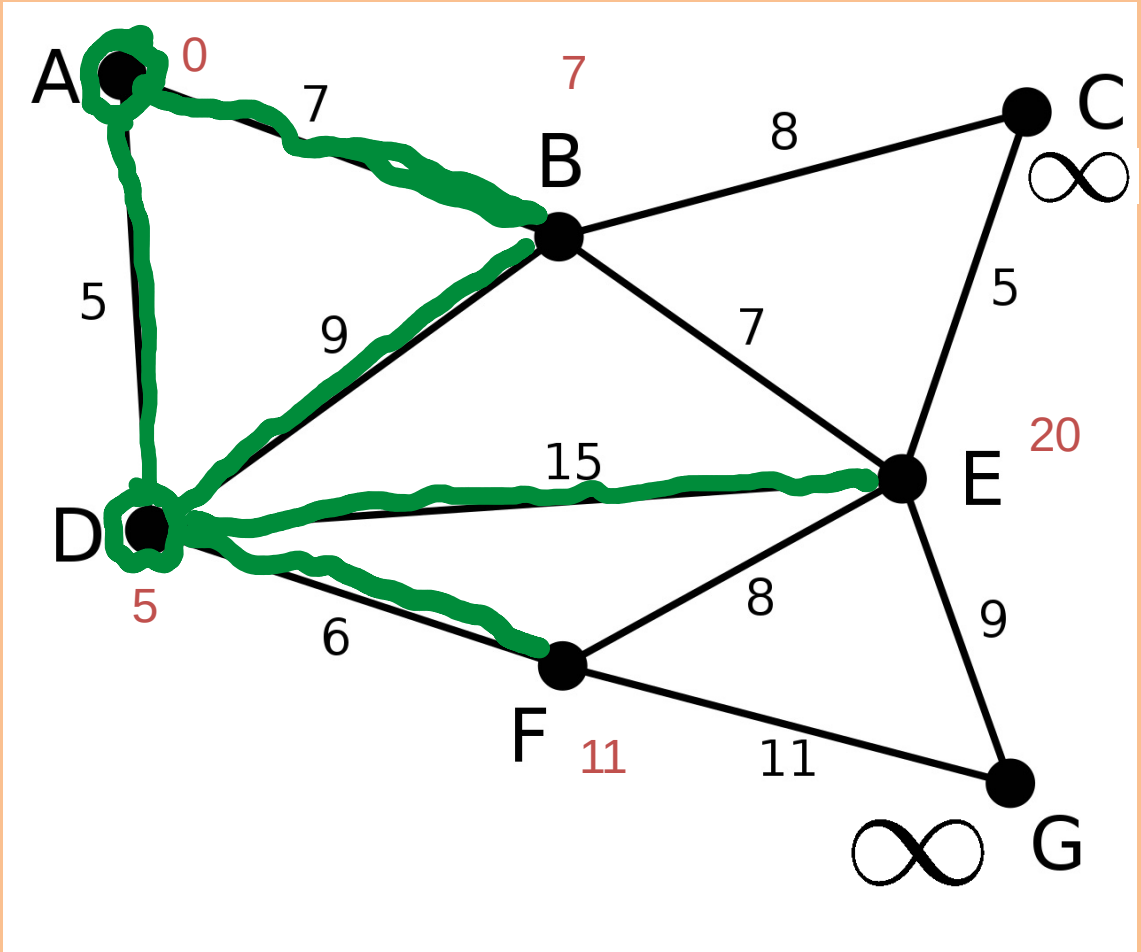
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



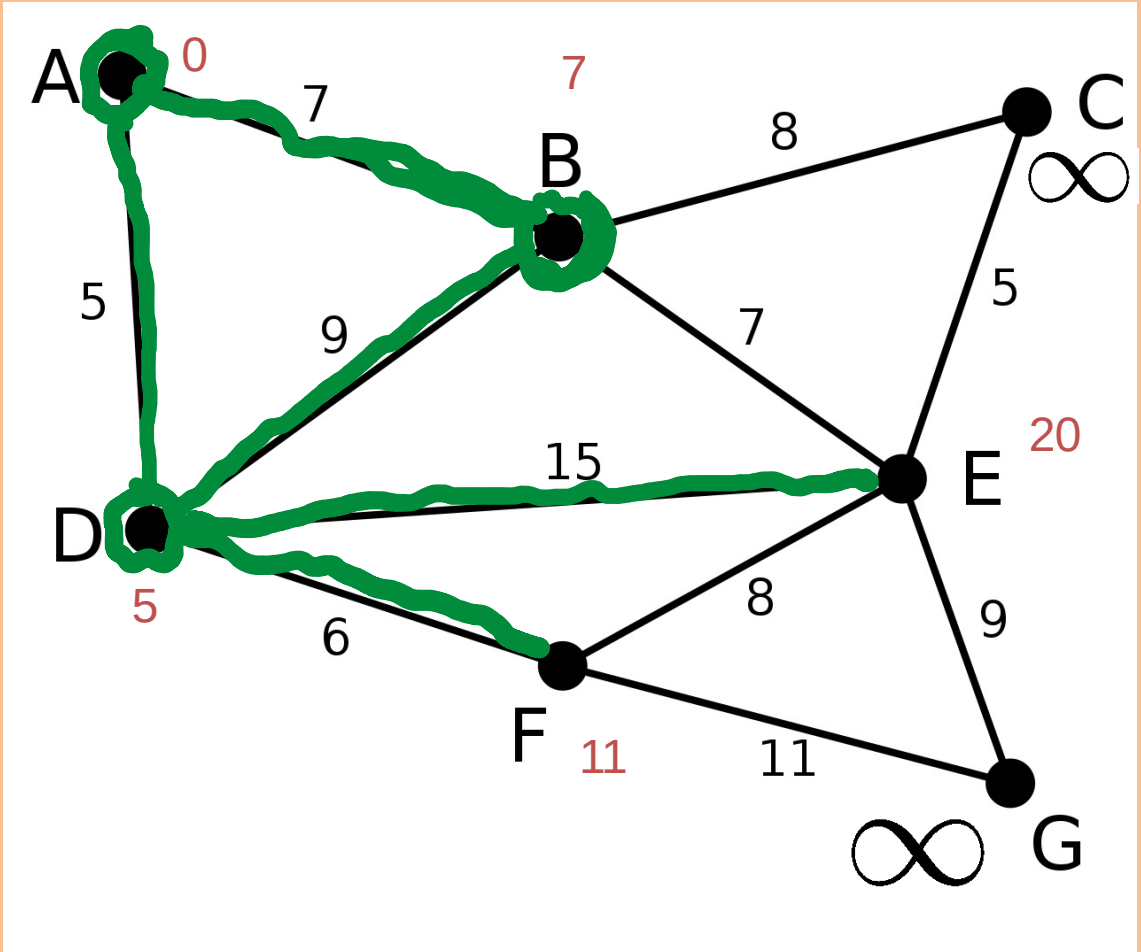
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



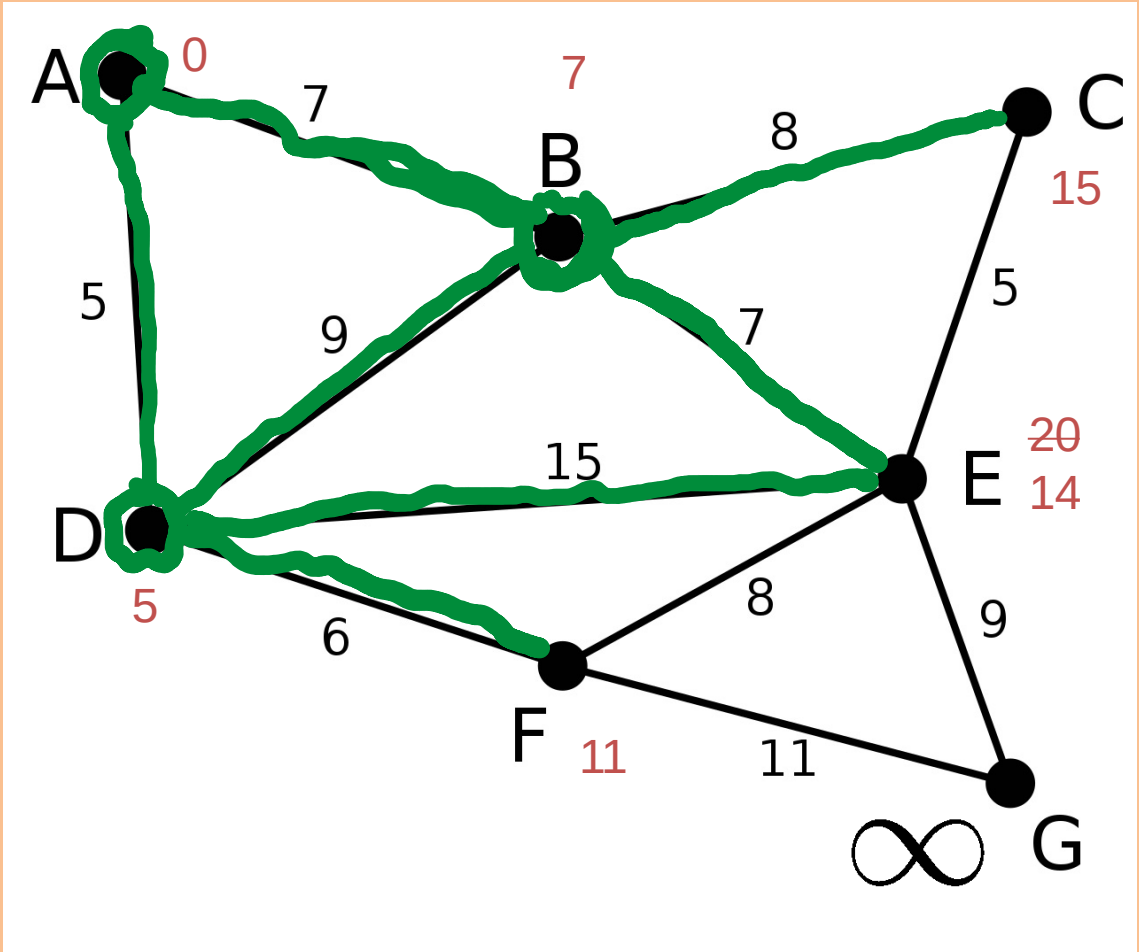
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



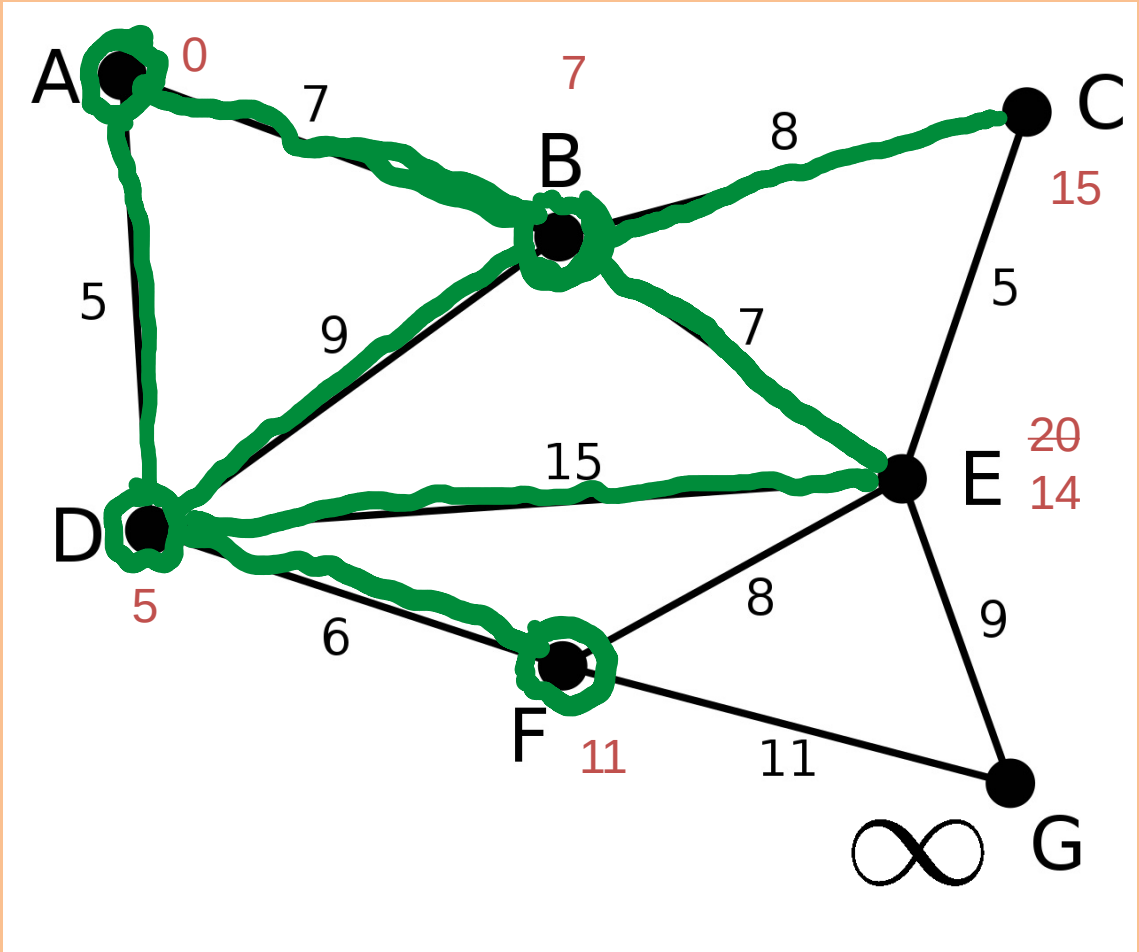
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



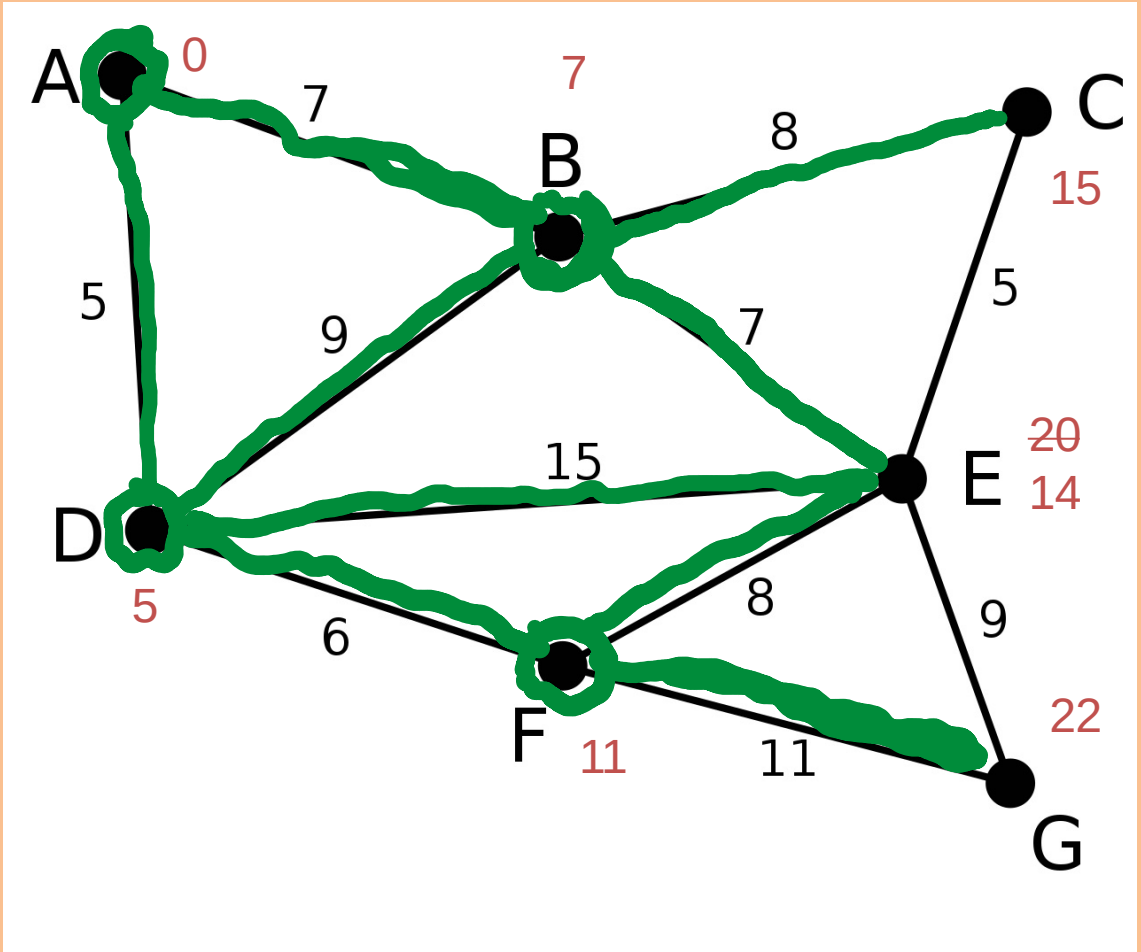
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



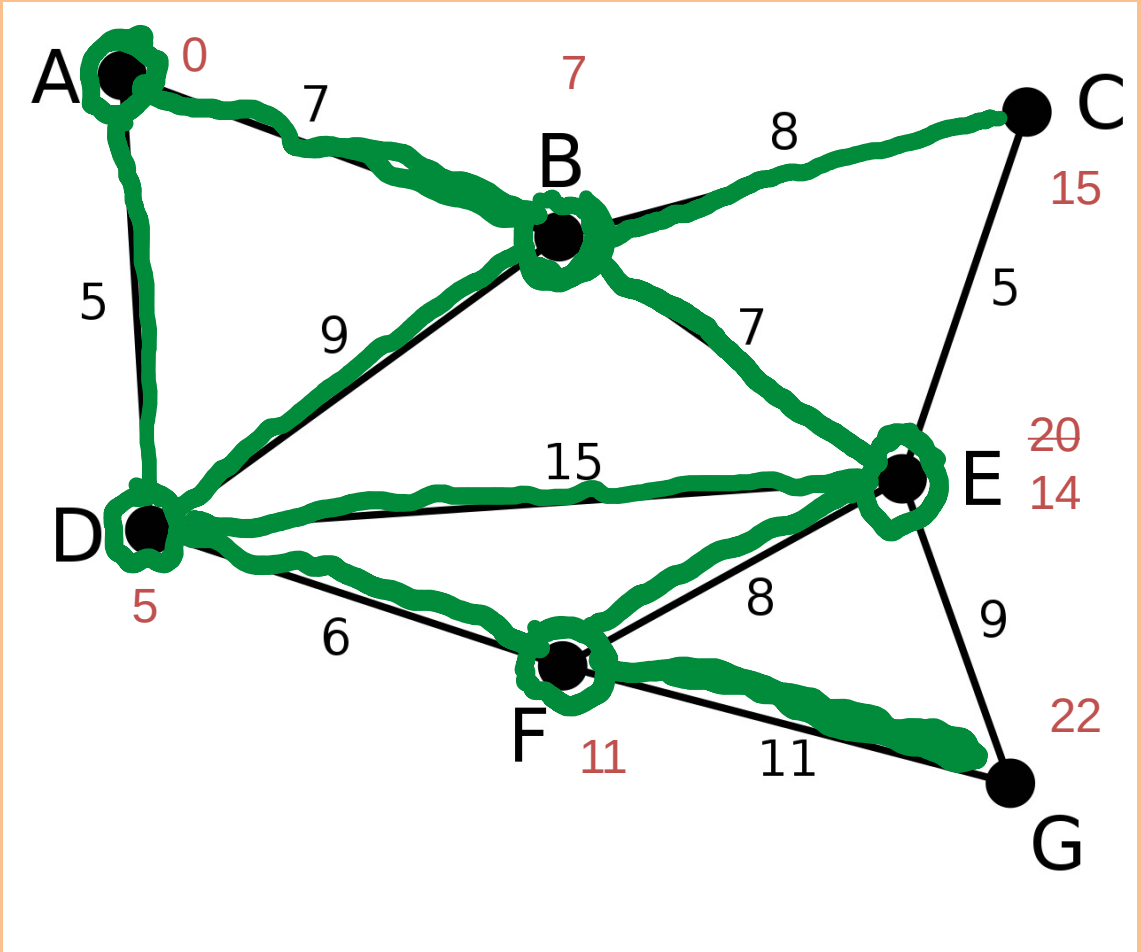
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



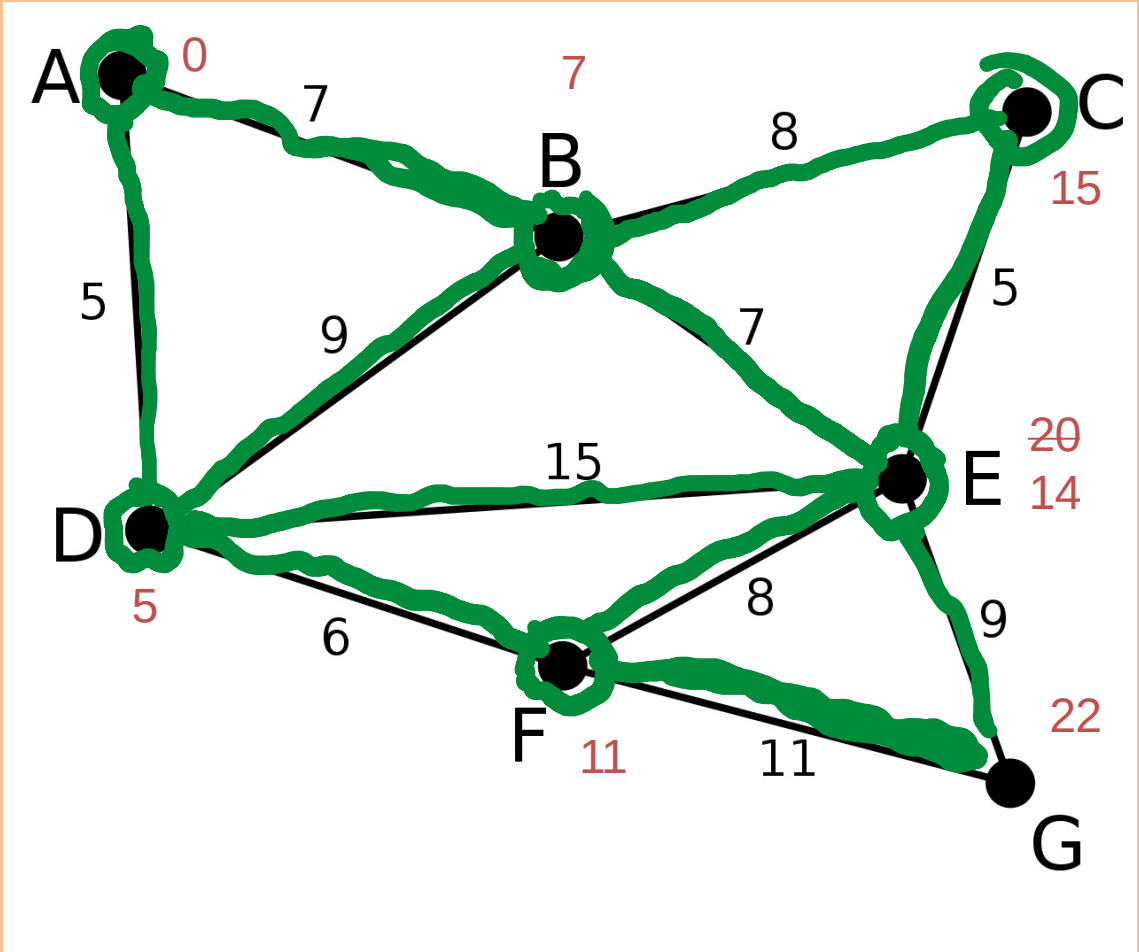
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



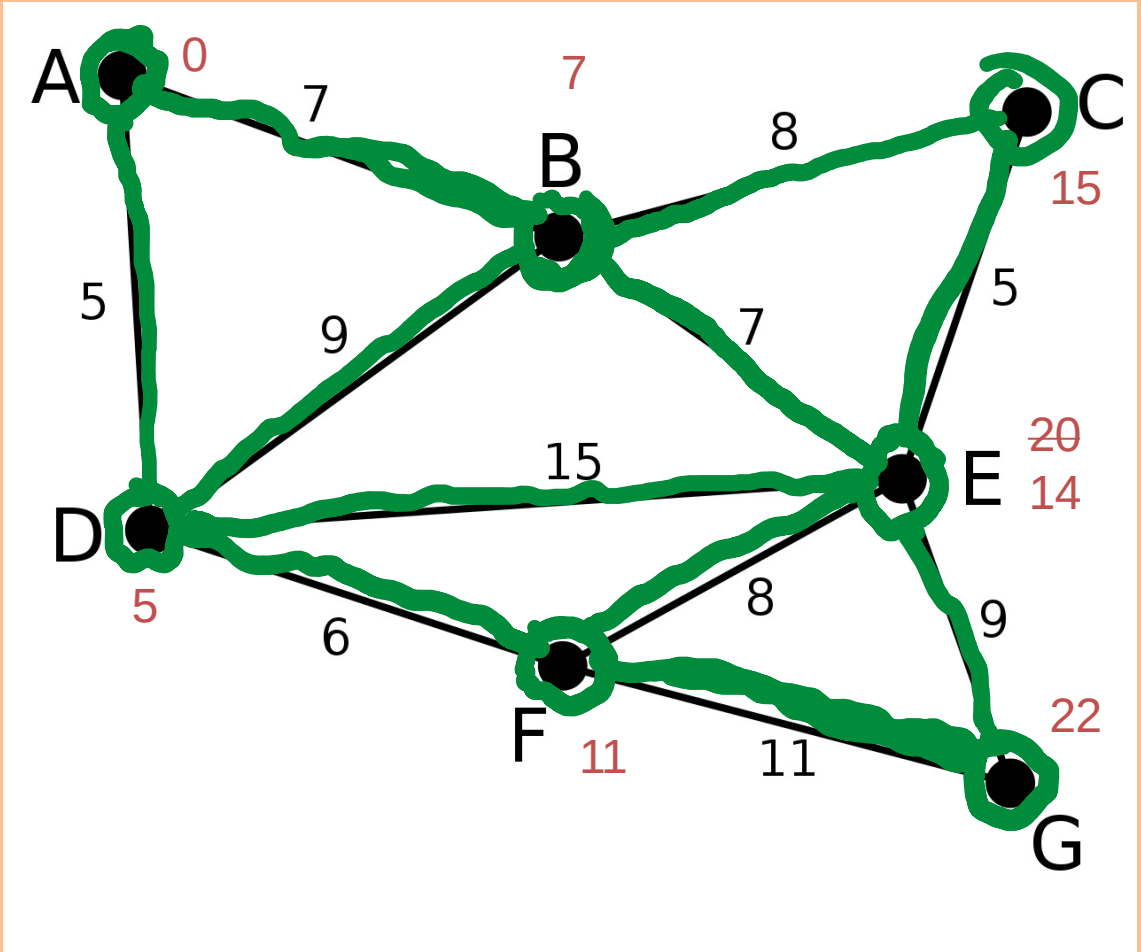
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



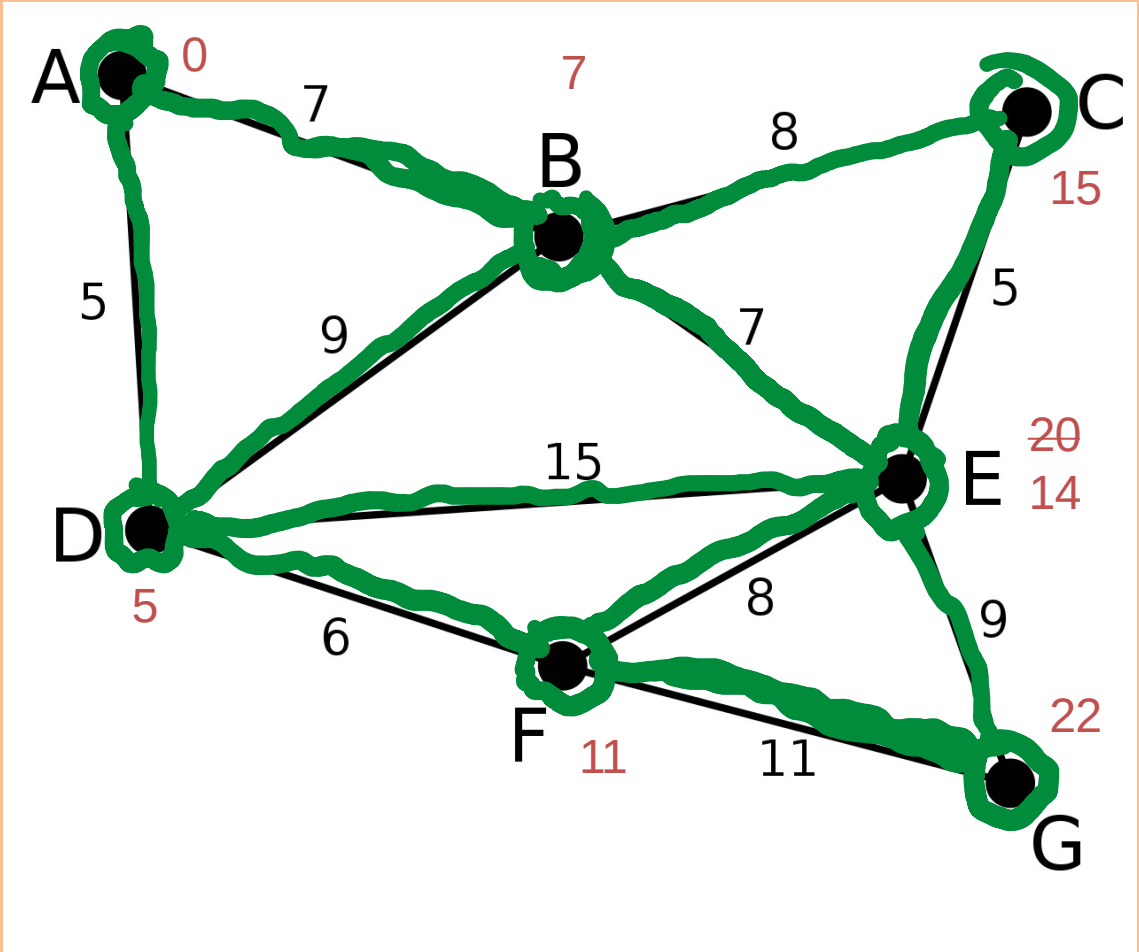
- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

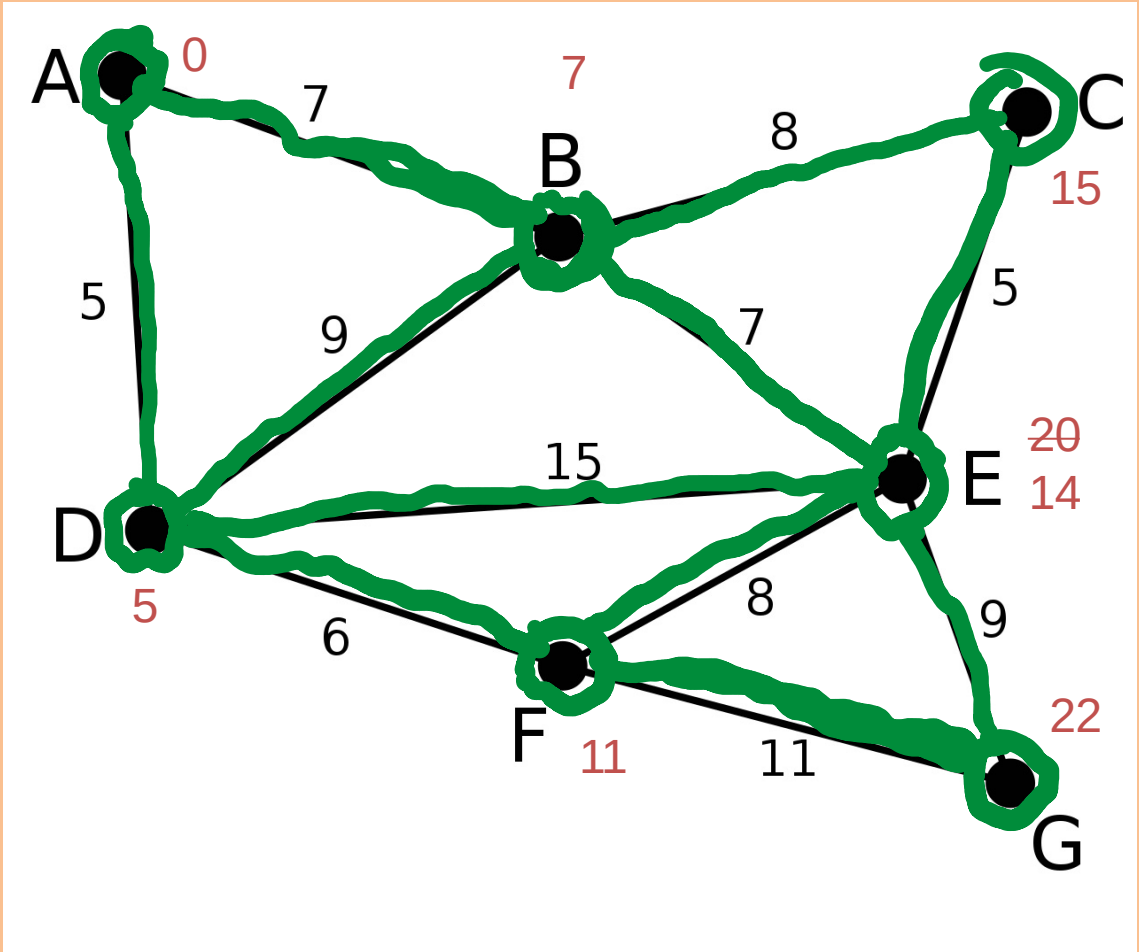
Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu



- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

- Nyní se od libovolného uzlu poskládá cesta k němu přes uzly s nejmenší hodnotou vzdálenosti od počátečního

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu

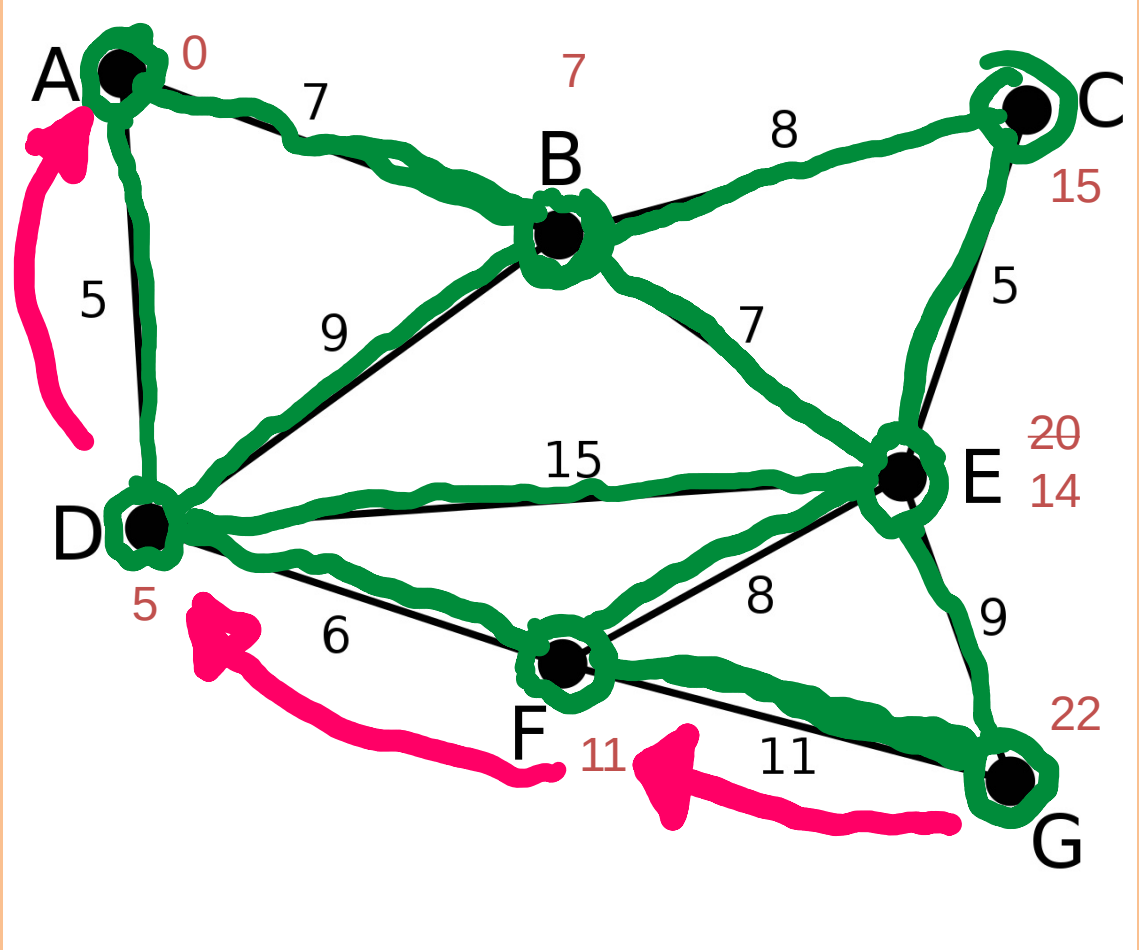


- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

- Nyní se od libovolného uzlu poskládá cesta k němu přes uzly s nejmenší hodnotou vzdálenosti od počátečního

- Např. A→G:

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu

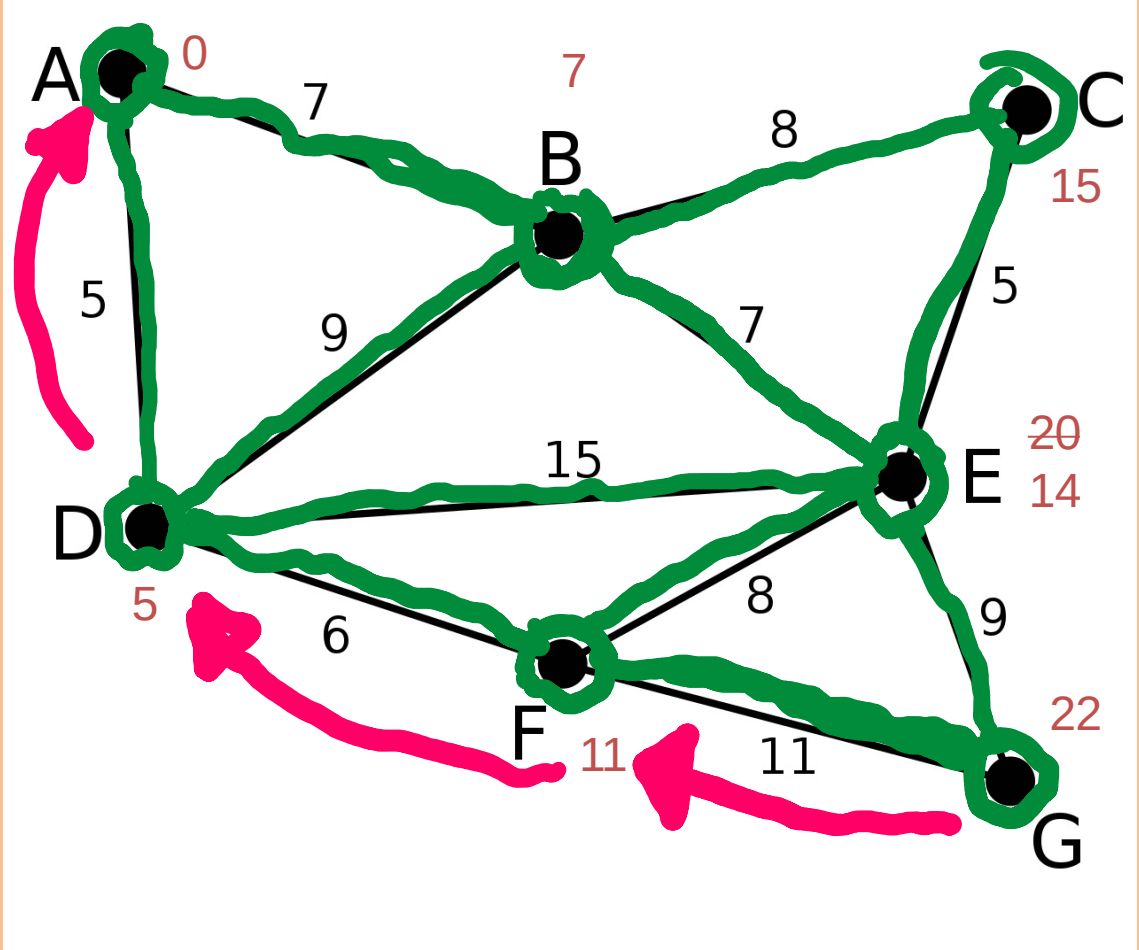


- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

- Nyní se od libovolného uzlu poskládá cesta k němu přes uzly s nejmenší hodnotou vzdálenosti od počátečního

- Např. A→G:

Neinformované prohledávací strategie: Dijkstrův algoritmus pro nejkratší cestu

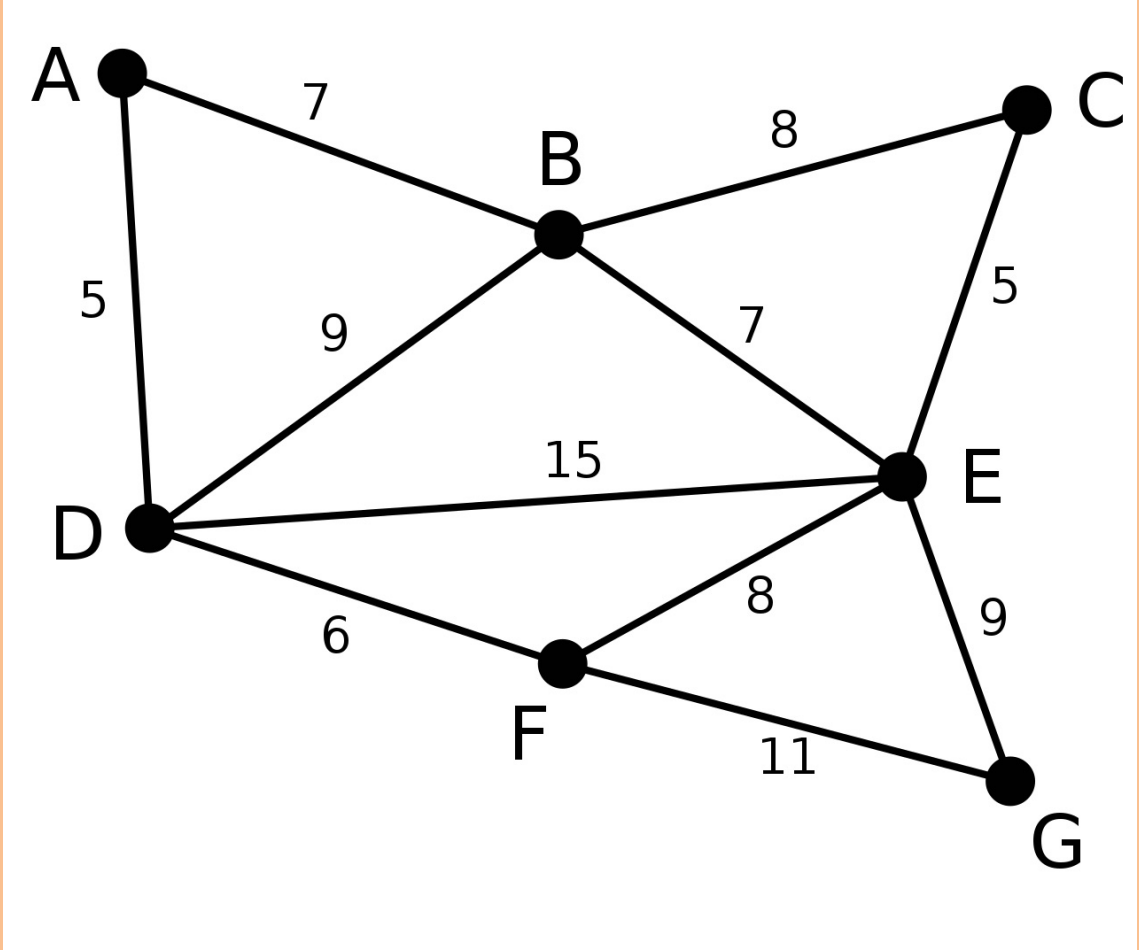


- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu (na počátku nekonečno s výjimkou počátku

- Nyní se od libovolného uzlu poskládá cesta k němu přes uzly s nejmenší hodnotou vzdálenosti od počátečního

- Např. A→G: A→D→F→G

Informovaná prohledávací strategie: A* pro nejkratší cestu



- Hrany označené jejich cenou
- Uzlům přiřazujeme vzdálenost od počátečního uzlu zvětšenou o odhad vzdálenosti k cíli:
- $f(n) = g(n)+h(n)$

- Heuristika $h(n)$ musí být pro úplnost funkce, která je *přípustná*, tj. nedává odhad větší, než je skutečnost (lze použít např. vzdušnou vzdálenost);
- nepřípustné heuristiky mohou činnost zrychlit, ale s rizikem, že některá řešení nenajdou.

Vyhledávání ve složitých prostředích

- Lokální hledání a optimalizace
- Hledání šplháním do kopce
- Stochastické šplhání, šplhání s náhodným restartem, simulované žíhání, jsou jediným zdrojem informací o prostředí, šplhání s místm reflektorem (kombinace výsledků z několika paralelních běhů), evoluční algoritmy
- Nedeterministická hledání
- Hledání v částečně pozorovatelném prostředí
- Hledání v neznámém prostředí a on-line aktéři zpracovávající vjemy okamžitě (a rizika slepých uliček)