DELTA TopGun

(14) Složité úlohy

Luboš Zápotočný, Tomáš Faltejsek, Michal Havelka

2023

Obsah

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Pokud pojedeme na výlet autem, budeme muset platit za benzín a budeme tedy chtít:

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Pokud pojedeme na výlet autem, budeme muset platit za benzín a budeme tedy chtít:

- co nejefektivněji projet jednotlivé památky

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Pokud pojedeme na výlet autem, budeme muset platit za benzín a budeme tedy chtít:

- co nejefektivněji projet jednotlivé památky
- s co nejmenší spotřebou benzínu

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Pokud pojedeme na výlet autem, budeme muset platit za benzín a budeme tedy chtít:

- co nejefektivněji projet jednotlivé památky
- s co nejmenší spotřebou benzínu
- vrátit se na konci výletu domů

Problém obchodního cestujícího

Mějme mapu České republiky a seznam památek, které bychom chtěli navštívit

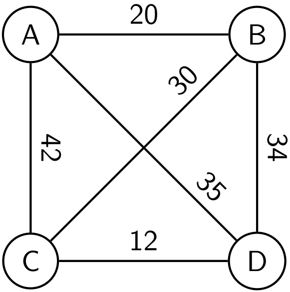
Máme také k dispozici vzdálenosti (po silnici) mezi jednotlivými místy

Pokud pojedeme na výlet autem, budeme muset platit za benzín a budeme tedy chtít:

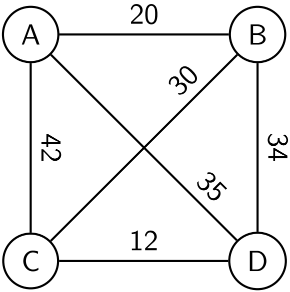
- co nejefektivněji projet jednotlivé památky
- s co nejmenší spotřebou benzínu
- vrátit se na konci výletu domů

Pizzérie chce pomocí motorky rozvést pět objednaných pizz v rámci jednoho výjezdu motorky

Ukázka

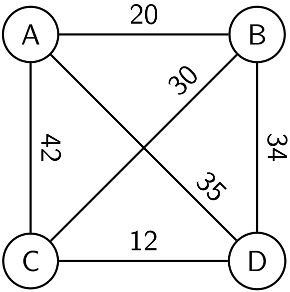


Ukázka



Pokud budeme začínat ve vrcholu A, jak bude vypadat nejoptimálnější cesta skrz všechny body?

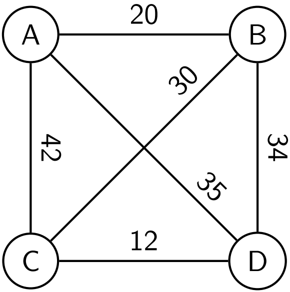
Ukázka



Pokud budeme začínat ve vrcholu A, jak bude vypadat nejoptimálnější cesta skrz všechny body?

- $A \to B \to D \to C \to A =$

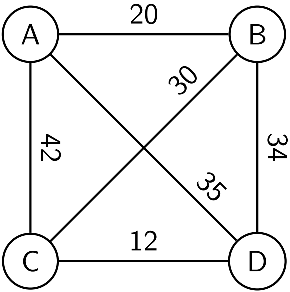
Ukázka



Pokud budeme začínat ve vrcholu A, jak bude vypadat nejoptimálnější cesta skrz všechny body?

- $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A = 20 + 34 + 12 + 42 = 108$
- $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A =$

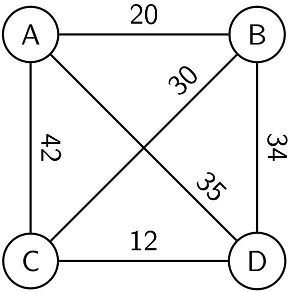
Ukázka



Pokud budeme začínat ve vrcholu A, jak bude vypadat nejoptimálnější cesta skrz všechny body?

- $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A = 20 + 34 + 12 + 42 = 108$
- $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A = 35 + 34 + 30 + 42 = 141$
- $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A =$

Ukázka



Pokud budeme začínat ve vrcholu A, jak bude vypadat nejoptimálnější cesta skrz všechny body?

- $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A = 20 + 34 + 12 + 42 = 108$
- $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A = 35 + 34 + 30 + 42 = 141$
- $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A = 20 + 30 + 12 + 35 = 97$
- ...

Problém obchodního cestujícího - bruteforce

Bruteforce

Problém obchodního cestujícího - bruteforce

Bruteforce

V případě, kdy máme možnost přejít z každého vrcholu do všech ostatních konstruujeme řetězce znaků

- A BDC A
- A DBC A
- A BCD A
- ...

Problém obchodního cestujícího - bruteforce

Bruteforce

V případě, kdy máme možnost přejít z každého vrcholu do všech ostatních konstruujeme řetězce znaků

- A BDC A
- A DBC A
- A BCD A
- ...

Mezi startovním a koncovým znakem A se nacházejí vždy 3 (neboli $4-1$) znaky a na první místo si můžeme vybrat z $n-1$ ostatních znaků

Problém obchodního cestujícího - bruteforce

Bruteforce

V případě, kdy máme možnost přejít z každého vrcholu do všech ostatních konstruujeme řetězce znaků

- A BDC A
- A DBC A
- A BCD A
- ...

Mezi startovním a koncovým znakem A se nacházejí vždy 3 (neboli $4-1$) znaky a na první místo si můžeme vybrat z $n-1$ ostatních znaků

Na následujícím místě už pouze $n-2$ (jeden znak jsme již použili) (do měst se již nevracíme)

Problém obchodního cestujícího - bruteforce

Bruteforce

V případě, kdy máme možnost přejít z každého vrcholu do všech ostatních konstruujeme řetězce znaků

- A BDC A
- A DBC A
- A BCD A
- ...

Mezi startovním a koncovým znakem A se nacházejí vždy 3 (neboli $4 - 1$) znaky a na první místo si můžeme vybrat z $n - 1$ ostatních znaků

Na následujícím místě už pouze $n - 2$ (jeden znak jsme již použili) (do měst se již nevracíme)

Poté máme už pouze $n - 3$ možností (v naší ukázce $n - 3 = 4 - 3 = 1$) a tak dále

Problém obchodního cestujícího - bruteforce

Dostáváme postupně hodnoty

- $n - 1$
- $n - 2$
- $n - 3$
- ...

které nám reprezentují počet možností výběru znaku

Problém obchodního cestujícího - bruteforce

Dostáváme postupně hodnoty

- $n - 1$
- $n - 2$
- $n - 3$
- ...

které nám reprezentují počet možností výběru znaku

Výběr je nezávislý na předchozím výběru a proto můžeme aplikovat kombinatorické pravidlo o násobení

Problém obchodního cestujícího - bruteforce

Dostáváme postupně hodnoty

- $n - 1$
- $n - 2$
- $n - 3$
- ...

které nám reprezentují počet možností výběru znaku

Výběr je nezávislý na předchozím výběru a proto můžeme aplikovat kombinatorické pravidlo o násobení

Dostáváme tedy složitost

$$(n-1) \times (n-2) \times (n-3) \times .... \times (2) \times (1) =$$

Problém obchodního cestujícího - bruteforce

Dostáváme postupně hodnoty

- $n - 1$
- $n - 2$
- $n - 3$
- ...

které nám reprezentují počet možností výběru znaku

Výběr je nezávislý na předchozím výběru a proto můžeme aplikovat kombinatorické pravidlo o násobení

Dostáváme tedy složitost

$$(n-1) \times (n-2) \times (n-3) \times .... \times (2) \times (1) = (n-1)!$$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe?

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!**

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! =$ $20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! =$ 20397882081197443358640281739902897356800000000

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

- $n = 7 \rightarrow 6272$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

- $n = 7 \rightarrow 6272$
- $n = 10 \rightarrow 102400$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

- $n = 7 \rightarrow 6272$
- $n = 10 \rightarrow 102400$
- $n = 15 \rightarrow 7372800$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! =$ $20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

- $n = 7 \rightarrow 6272$
- $n = 10 \rightarrow 102400$
- $n = 15 \rightarrow 7372800$
- $n = 40 \rightarrow 1759218604441600$

Problém obchodního cestujícího - bruteforce

Jak velké instance tohoto problému dokážeme tedy zpracovat?

- $(5-1)! = 24$
- $(7-1)! = 720$
- $(10-1)! = 362880$
- $(15-1)! = 87178291200$
- $(20-1)! = 121645100408832000$
- $(40-1)! = 20397882081197443358640281739902897356800000000$

Dokážeme to lépe? **ANO!** (na příští přednášce)

Ukážeme si postup, který tento problém bude řešit v čase $\mathcal{O}(n^2 2^n)$

- $n = 7 \rightarrow 6272$
- $n = 10 \rightarrow 102400$
- $n = 15 \rightarrow 7372800$
- $n = 40 \rightarrow 1759218604441600$

Opakování logických výrazů

- $\neg$

Opakování logických výrazů

- $\neg$ negace

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A
1	1	
1	0	
0	1	
0	0	

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A	$\wedge$
1	1	0	
1	0	0	
0	1	1	
0	0	1	

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A	$\wedge$	$\vee$
1	1	0	1	
1	0	0	0	
0	1	1	0	
0	0	1	0	

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A	$\wedge$	$\vee$	$\implies$
1	1	0	1	1	
1	0	0	0	1	
0	1	1	0	1	
0	0	1	0	0	

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A	$\wedge$	$\vee$	$\implies$	$\iff$
1	1	0	1	1	1	
1	0	0	0	1	0	
0	1	1	0	1	1	
0	0	1	0	0	1	

Opakování logických výrazů

- $\neg$ negace - !(...)
- $\wedge$ konjunkce - (...) && (...)
- $\vee$ disjunkce - (...) || (...)
- $\implies$ implikace
- $\iff$ ekvivalence

A	B	$\neg$A	$\wedge$	$\vee$	$\implies$	$\iff$
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Booleovské formule

Konjunktivní normální forma

Booleovské formule

Konjunktivní normální forma - „konjunkce skupin disjunkcí“

Booleovské formule

Konjunktivní normální forma - „konjunkce skupin disjunkcí“

$(A \vee \neg B) \wedge (B \vee C \vee D) \wedge (\neg A \vee D) \wedge (B \vee D) \wedge (A)$

Booleovské formule

Konjunktivní normální forma - „konjunkce skupin disjunkcí“

$(A \vee \neg B) \wedge (B \vee C \vee D) \wedge (\neg A \vee D) \wedge (B \vee D) \wedge (A)$

Ohodnocení booleovské formule je přiřazení logických hodnot (True, False) jednotlivým proměnným

Booleovské formule

Konjunktivní normální forma - „konjunkce skupin disjunkcí"

$(A \vee \neg B) \wedge (B \vee C \vee D) \wedge (\neg A \vee D) \wedge (B \vee D) \wedge (A)$

Ohodnocení booleovské formule je přiřazení logických hodnot (True, False) jednotlivým proměnným

Například

- A = True
- B = False
- C = False
- D = True

Booleovské formule

Konjunktivní normální forma - „konjunkce skupin disjunkcí“

$(A \vee \neg B) \wedge (B \vee C \vee D) \wedge (\neg A \vee D) \wedge (B \vee D) \wedge (A)$

Ohodnocení booleovské formule je přiřazení logických hodnot (True, False) jednotlivým proměnným

Například

- A = True
- B = False
- C = False
- D = True

Takto nastavené přiřazení hodnot splňuje celou formuli a výsledný logický výrok je pravdivý (True)

Splnitelnost booleovské formule

Problém splnitelnosti booleovské formule je problém nalezení splňujícího ohodnocení booleovské formule

Splnitelnost booleovské formule

Problém splnitelnosti booleovské formule je problém nalezení splňujícího ohodnocení booleovské formule

Každá proměnná může nabývat dvou hodnot (True, False)

Splnitelnost booleovské formule

Problém splnitelnosti booleovské formule je problém nalezení splňujícího ohodnocení booleovské formule

Každá proměnná může nabývat dvou hodnot (True, False)

Ohodnocení proměnných (pole booleovských hodnot) můžeme v lineárním čase ověřit

Splnitelnost booleovské formule

Problém splnitelnosti booleovské formule je problém nalezení splňujícího ohodnocení booleovské formule

Každá proměnná může nabývat dvou hodnot (True, False)

Ohodnocení proměnných (pole booleovských hodnot) můžeme v lineárním čase ověřit

Jedná se o konjunkce klausulí, stačí tedy iterovat přes jednotlivé klausule a provádět:

Splnitelnost booleovské formule

Problém splnitelnosti booleovské formule je problém nalezení splňujícího ohodnocení booleovské formule

Každá proměnná může nabývat dvou hodnot (True, False)

Ohodnocení proměnných (pole booleovských hodnot) můžeme v lineárním čase ověřit

Jedná se o konjunkce klausulí, stačí tedy iterovat přes jednotlivé klausule a provádět:

- dosadit do klauzule dané hodnoty proměnných
- vyhodnotit výraz
- pokud je pravdivý, pokračovat
- pokud je nepravdivý, ukončit algoritmus s chybou

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Máme dvě možnosti, jak nastavit hodnotu poslední proměnné x_n

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Máme dvě možnosti, jak nastavit hodnotu poslední proměnné x_n

Předposlední proměnné x_{n-1} můžeme také nastavit dvě různé hodnoty

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Máme dvě možnosti, jak nastavit hodnotu poslední proměnné x_n

Předposlední proměnné x_{n-1} můžeme také nastavit dvě různé hodnoty

Výsledné kombinace předposlední a poslední proměnné jsou ale již čtyři

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Máme dvě možnosti, jak nastavit hodnotu poslední proměnné x_n

Předposlední proměnné x_{n-1} můžeme také nastavit dvě různé hodnoty

Výsledné kombinace předposlední a poslední proměnné jsou ale již čtyři

- 0 0
- 0 1
- 1 0
- 1 1

Problém splnitelnosti booleovské formule - bruteforce

Mějme n proměnných, které se ve formuli vyskytují

Každá proměnná může nabývat dvou hodnot (True, False)

Máme dvě možnosti, jak nastavit hodnotu poslední proměnné x_n

Předposlední proměnné x_{n-1} můžeme také nastavit dvě různé hodnoty

Výsledné kombinace předposlední a poslední proměnné jsou ale již čtyři

- 0 0
- 0 1
- 1 0
- 1 1

Přidáním $n-2$ proměnné a jejích dvou možných hodnot již dostáváme osm kombinací

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 =$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$
- $m = 50, n = 10 \rightarrow 51200$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$
- $m = 50, n = 10 \rightarrow 51200$
- $m = 50, n = 15 \rightarrow 1638400$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$
- $m = 50, n = 10 \rightarrow 51200$
- $m = 50, n = 15 \rightarrow 1638400$
- $m = 50, n = 40 \rightarrow 54975581388800$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$
- $m = 50, n = 10 \rightarrow 51200$
- $m = 50, n = 15 \rightarrow 1638400$
- $m = 50, n = 40 \rightarrow 54975581388800$
- $m = 50, n = 50 \rightarrow 56294995342131200$

Problém splnitelnosti booleovské formule - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu splnitelnosti formule

Pokud si označíme m jako počet literálů (výskyt nějaké proměnné ve formuli) můžeme celkovou časovou složitost odhadnout $\mathcal{O}(m2^n)$

například pro instance s těmito parametry

- $m = 50, n = 5 \rightarrow 1600$
- $m = 50, n = 10 \rightarrow 51200$
- $m = 50, n = 15 \rightarrow 1638400$
- $m = 50, n = 40 \rightarrow 54975581388800$
- $m = 50, n = 50 \rightarrow 56294995342131200$
- $m = 100, n = 100 \rightarrow 126765060022822940149670320537600$

Plnění batohu

Mějme batoh s omezením na maximální zatížení W

Plnění batohu

Mějme batoh s omezením na maximální zatížení W

Mějme množinu věcí, které bysme do batohu mohli dát

Plnění batohu

Mějme batoh s omezením na maximální zatížení W

Mějme množinu věcí, které bysme do batohu mohli dát

Každá věc má svoji váhu $w(x)$ a cenu $c(x)$

Plnění batohu

Mějme batoh s omezením na maximální zatížení W

Mějme množinu věcí, které bysme do batohu mohli dát

Každá věc má svoji váhu $w(x)$ a cenu $c(x)$

Snažíme se naplnit batoh tak, aby jsme ho nepřetížili

Plnění batohu

Mějme batoh s omezením na maximální zatížení W

Mějme množinu věcí, které bysme do batohu mohli dát

Každá věc má svoji váhu $w(x)$ a cenu $c(x)$

Snažíme se naplnit batoh tak, aby jsme ho nepřetížili

Také dbáme na součet cen věcí, které se v batohu nalézají a snažíme se tento součet maximalizovat

Plnění batohu

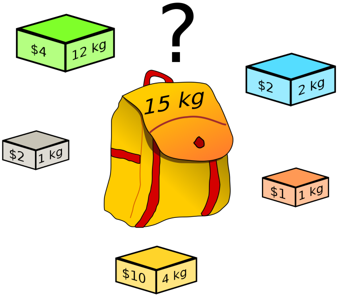
Mějme batoh s omezením na maximální zatížení W

Mějme množinu věcí, které bysme do batohu mohli dát

Každá věc má svoji váhu $w(x)$ a cenu $c(x)$

Snažíme se naplnit batoh tak, aby jsme ho nepřetížili

Také dbáme na součet cen věcí, které se v batohu nalézají a snažíme se tento součet maximalizovat



Plnění batohu - bruteforce

Podobný algoritmus jako na problém splnitelnosti Booleovské formule

Plnění batohu - bruteforce

Podobný algoritmus jako na problém splnitelnosti Booleovské formule

Mějme n věcí, které můžeme do batohu dát

Plnění batohu - bruteforce

- Podobný algoritmus jako na problém splnitelnosti Booleovské formule
- Mějme *n* věcí, které můžeme do batohu dát
- Každá věc může v batohu být nebo nebýt (True, False)

Plnění batohu - bruteforce

Podobný algoritmus jako na problém splnitelnosti Booleovské formule

Mějme *n* věcí, které můžeme do batohu dát

Každá věc může v batohu být nebo nebýt (True, False)

Máme dvě možnosti, jestli věc x_n do batohu dát, nebo ne

Plnění batohu - bruteforce

- Podobný algoritmus jako na problém splnitelnosti Booleovské formule
- Mějme n věcí, které můžeme do batohu dát
- Každá věc může v batohu být nebo nebýt (True, False)
- Máme dvě možnosti, jestli věc x_n do batohu dát, nebo ne
- Předposlední věc x_{n-1} můžeme do batohu dát, nebo také ne

Plnění batohu - bruteforce

- Podobný algoritmus jako na problém splnitelnosti Booleovské formule
- Mějme n věcí, které můžeme do batohu dát
- Každá věc může v batohu být nebo nebýt (True, False)
- Máme dvě možnosti, jestli věc x_n do batohu dát, nebo ne
- Předposlední věc x_{n-1} můžeme do batohu dát, nebo také ne
- Výsledné kombinace předposledního a posledního vložení do batohu jsou již čtyři

Plnění batohu - bruteforce

Podobný algoritmus jako na problém splnitelnosti Booleovské formule

Mějme n věcí, které můžeme do batohu dát

Každá věc může v batohu být nebo nebýt (True, False)

Máme dvě možnosti, jestli věc x_n do batohu dát, nebo ne

Předposlední věc x_{n-1} můžeme do batohu dát, nebo také ne

Výsledné kombinace předposledního a posledního vložení do batohu jsou již čtyři

- 0 0
- 0 1
- 1 0
- 1 1

Plnění batohu - bruteforce

Podobný algoritmus jako na problém splnitelnosti Booleovské formule

Mějme n věcí, které můžeme do batohu dát

Každá věc může v batohu být nebo nebýt (True, False)

Máme dvě možnosti, jestli věc x_n do batohu dát, nebo ne

Předposlední věc x_{n-1} můžeme do batohu dát, nebo také ne

Výsledné kombinace předposledního a posledního vložení do batohu jsou již čtyři

- 0 0
- 0 1
- 1 0
- 1 1

Přidáním $n - 2$ věci do batohu (dalších dvou možností) již dostáváme osm kombinací

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 =$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$
- $n = 10 \rightarrow 10240$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$
- $n = 10 \rightarrow 10240$
- $n = 15 \rightarrow 491520$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times ... \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$
- $n = 10 \rightarrow 10240$
- $n = 15 \rightarrow 491520$
- $n = 40 \rightarrow 43980465111040$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$
- $n = 10 \rightarrow 10240$
- $n = 15 \rightarrow 491520$
- $n = 40 \rightarrow 43980465111040$
- $n = 50 \rightarrow 56294995342131200$

Plnění batohu - bruteforce

Postupným dosazování dostáváme výpočet tohoto tvaru

$2 \times 2 \times \ldots \times 2 \times 2 \times 2 = 2^n$

Pro každou kombinaci musíme spustit lineární algoritmus pro kontrolu váhového limitu a pro výpočet ceny věcí v batohu

Pokud si označíme n jako počet věcí, které můžeme do batohu dát a každá věc má váhu váhu $w(x)$ a cenu $c(x)$ můžeme celkovou časovou složitost odhadnout $\mathcal{O}(n2^n)$

například pro instance s těmito parametry

- $n = 5 \rightarrow 160$
- $n = 10 \rightarrow 10240$
- $n = 15 \rightarrow 491520$
- $n = 40 \rightarrow 43980465111040$
- $n = 50 \rightarrow 56294995342131200$
- $n = 100 \rightarrow 126765060022822940149670320537600$