Randomizované algoritmy

I.Kolingerová

Obsah: 1. Motivace
2. Randomizované algoritmy
3. Příklady
4. Náhodná čísla
5. Závěr

1. Motivace

Př.: Pole prvků, polovina znak „a", polovina „b", pořadí neznámé, najděte „a".

Deterministicky: n/2 kroků v nejhorším případě (pokud napřed všechna „b")

Randomizovaně: kontrola v náhodném pořadí, s velkou pravděpodobností najdeme „a" brzy bez ohledu na typ vstupu (to je tzv. Las Vegas algoritmus)

2

Motivace

- Randomizovaný alg. může být jednodušší než deterministický při stejném nejhorším případě jako deterministický (Př.: quicksort)
- Jednoduchou determin.heuristiku s „dobrými výsledky" může být možné převést na algoritmus s dobrým nejhorším případem dodáním náhodnosti (př.:medián)
- Někdy náhodnost může řešit věci, kt. deterministicky nelze (problémy řešené orákulem) nebo se zdají nemožné

3

Motivace

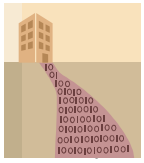
- Randomizace - vhodný nástroj pro vylepšování algoritmů, kt. mají špatný nejhorší případ a dobrý průměrný případ; nejhorší případ zůstává, ale závisí na tom, zda máme štěstí :-) , ne na pořadí vstupních dat

4

Motivace

Zvláště užitečné, pokud „zlomyslný protivník" se snaží algoritmu podstrčit „špatný" vstup ☺

Náhodnost všudypřítomná v kryptografii – pseudonáhodná čísla zde nelze, jsou predikovatelná, vlastně tak vznikne deterministický alg. (Zde nutný buď zdroj skutečně náhodných čísel nebo tzv. kryptograficky bezpečný generátor pseudonáh. čísel)



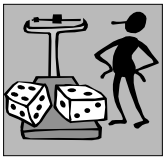
Další kritická aplikace – kvantové výpočty

Běžné aplikace – hry, simulace, numerika (randomizovaná aproximace)

2. Randomizované algoritmy

- Dělá nedeterministická, náhodná rozhodnutí – např. uvažuje vstupní data v náhodném pořadí, vybírá alternativu podle prahování na základě náhodného čísla apod.
- V určitém momentu dělá náhodný výběr
  - inkrementální - náhodné pořadí vkládání
  - D&C - náhodný výběr vzorku
- Obvykle jednodušší implementace, často v praxi rychlejší

6



Motivace

- Není jasné, zda vždy možný převod randomizovaného alg. na nerandomiz. bez zhoršení výkonu
- Není znám žádný randomiz. polynomiální alg. pro NP-úplné problémy
- Existují problémy, kde znám účinný randomiz. alg., ale není znám účinný deterministický alg., a není známo, zda jde o P nebo NP problém (např. test prvočísel)
- => někdy dá velmi praktické a mnohem jednodušší řešení než deterministický

7

Motivace

- V posledních letech velmi populární
- Někdy naopak derandomizace
- Někdy v kombinaci s přibliž. řešením pro NP-úplné problémy

8

Randomizované algoritmy

a) Randomizované inkrementální algoritmy
- Lineární programování
- Konstrukce různých dat.struktur (např. stromů), které nechceme vyvažovat a nechceme záviset na případném uspořádání vstup. dat

b) Rozděl a panuj
- Náhodné vzorkování
- Náhodná volba pivotu pro dělicí krok D&C

9

Lineární programování

Minimaliz. $c_1x_1 + c_2x_2 + \ldots + c_dx_d$

s omezeními
$$b_{1,1}x_1 \ldots + b_{1,d}x_d \le b_{1,d+1}$$
$$b_{2,1}x_1 \ldots + b_{2,d}x_d \le b_{2,d+1}$$
$$\ldots\ldots$$
$$b_{n,1}x_1 \ldots + b_{n,d}x_d \le b_{n,d+1}$$

kde c_i, $b_{i,j}$ jsou reálná čísla (vstup)

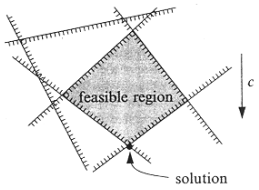
Omezení - poloprostory v E^d, dosaž. oblast - průnik n poloprostorů = konvexní polyhedron

Hledáme bod v polyhedronu, kt. je extrémem ve směru specif. cenovou funkcí

Přístup vhodný pro d malé, konst. (2,3)

10

Lineární programování



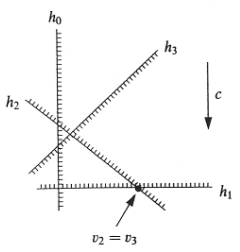
- Původní technika: Megiddo and Dyer; randomiz. verze jednodušší a lepší složitost
- Př. v 2D: transformovat na minimaliz. fce c:($x1$,$x2$) -> $x2$, hledáme nejnižší bod v dosaž. oblasti
- Přidat $x1 \geq 0$, $x2 \geq 0$ (=>ohraničenost řešení)

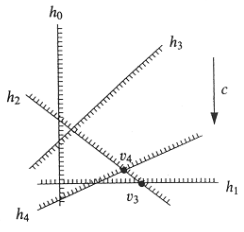
11

Lineární programování

Randomiz. inkrement. vkládání: vkládání polorovin po jedné v náh. pořadí, udržuje se optimum v_i

Nově vložená polorovina h_{i+1} buď optimum nezmění nebo extrém. vrchol odřízne, nový extrém. vrchol v_{i+1} nalezneme určováním průsečíků s vkládanou přímkou - vlastně 1D lineár. programování na přímce, lineár. čas





12

Lineární programování

- Složitost na 1 polorovinu: Pr[v_{i-1} je prvkem h_i]O(1) + Pr[v_{i-1} není prvkem h_i]O(i) , po úpravě celkově O(1) => celkem O(n) na celý problém
- Pro R^d: O($d!n$)
  - lineární, pokud d konst.
  - výhodné, pokud d malé

13

Vzorkování

- Nezjišťujeme hodnotu pro každou osobu nebo množinu jednotek, ale jen pro některé
  - náhodné
  - nenáhodné
- Důvody: čas a peníze (např. potřeba trénovaných dotazujících, výrobce aut – nemůže u všech testovat odolnost proti srážce
- Ze vzorků odhady pro celou populaci, vzorek musí být dost velký (reprezentativnost) i dost malý (použitelnost)

14

Náhodné vzorkování

- U všech položek stejná pravděpodobnost výběru, lze ji spočítat
- Není zkreslení vlivem výběru jednotek do vzorku
- Typy náh. vzorkování:
  - jednoduché
  - systematické
  - stratifikované
  - shlukované
  - víceúrovňové

15

Náhodné vzorkování

- Použití v D&C - pro dělení použita náhodně vybraná podmnožina objektů (viz quicksort)
- Často pro vyhledávací datové struktury se vzorkem velkých dat
- Užití často pro přibližná řešení na náhodném vzorku dat (např. mediány)
- Též vhodné pro paralelizaci

16

Jednoduché náhodné vzorkování

- Každá položka populace má stejnou šanci na zařazení do výběru
- Př.: výběr x jmen z telefonního seznamu – očíslovat a vybrat náhodně s rovnoměr. pravděpodobností
- Př.: 6 čísel z 49 ve Sportce
- Jednoduché, pro malé aplikace snadné, pro velké problematické – nutno seznam všech položek a očíslovat

17

Systematické vzorkování

- Též "intervalové vzorkování"
- Mezi výběry interval
- 1. vybrána náhodně
- Často v průmyslu nebo obchodu
- Př.:výběr položky z linky po 15 min. , výběr každé 20.položky, každý 10. návštěvník, každý 5.dům ...
- Chceme-li pevnou velikost vzorku: I = N/n (I-interval, N-velikost populace, n-velikost vzorku); zaokrouhlit

18

Systematické vzorkování

- Př.: systematický vzorek 500 studentů z 10tis. na univerzitě – $I=10^4/500=20$, všem studentům čísla, poč. bod: náh. číslo 1-20, např. 9
- Vzorek: 9, 29, 49, ..., 9969, 9989

- Výhody:
  - jednodušší je vybrat 1 náh. číslo než spoustu
  - dobré rozložení v populaci
- Nevýhody:
  - počát. seznam
  - úchylky ve struktuře dat => zkreslení výsledků

19

Stratifikované vzorkování

- Při náh. vzorkování možnost zmeškání určité skupiny
- Sdružit populaci do skupin a vzorkovat v každé skupině
- Skupiny – strata (stát, věk, pohlaví, rodinný stav)
- Př.: Reakce studentů univerzity na nějaké nové opatření – strata – ročníky, uvnitř každého jednoduchý náhodný nebo systematický vzorek ze studentů
- Vhodné, pokud skupiny jednoduché, dobře pozorovatelné, uvnitř skupin podobný názor
- Nutno vzít v potaz proporce skupin

20

Shlukované vzorkování

- Rozšíření vzorků v celé populaci někdy drahé nebo obtížné => rozdělit populaci na náhodný počet shluků, do vzorku zahrnout všechny jednotky vybraných shluků
- Z nevybraných shluků se nebere nic (odlišnost od stratif.)
- Př. shluků: továrny, školy, geograf. oblasti
- Př.: Zjistit v rámci celé ČR, které sporty pěstují desetiletí žáci – vybrat náhodně 100 škol v celé zemi, z těchto shluků vybrat všechny.

21

Shlukované vzorkování

- Výhody: redukce ceny, jednodušší, vzorek lokalizován v několika shlucích
- Nevýhody: méně přesné výsledky, větší vzork. chyba než při stejném vzorku a náhod. vzorkování

22

Víceúrovňové vzorkování

- Jako shlukové, ale zahrnuje výběr vzorku z každého vybraného shluku (shluk se npoužije celý)
- Výběr vzorků na nejméně 2 úrovních, v 1. zvoleny velké skupiny nebo shluky, v 2. jednotky z vybraných shluků

Př.: Volební okrsky – shluky, z nich bloky domů, domy

- Výhody:
  - pohodlné, ekonomické, účinné
  - nejsou nutné úplné seznamy cíl. populace – levnější
- Nevýhoda: malá přesnost díky vyšší vzork. chybě

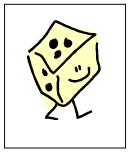
23

Jiné dělení randomizovaných algoritmů

- Monte Carlo
- Las Vegas

24

Monte Carlo



- Vždy dostaneme odpověď, ne vždy správnou
- Typicky zde náhodnost užita pro vedení algoritmu k řešení rychlejším způsobem s rizikem chyby
- Nelze určit, zda odpověď algoritmu je správná, ale můžeme pustit opakovaně a srovnat výsledky
- Výstup lze chápat jako náh. proměnnou

25

Monte Carlo

Př.: Rozhodněte, zda *n* je prvočíslo nebo číslo složené

Využije se hledání dělitele – hledá se *a* z intervalu 2...*n*/2 takové, že *n* je dělitelné *a*

Randomiz. algoritmus:

- vzorkuje 10 čísel z daného intervalu, podle toho odpoví

26

Las Vegas

- Vždy správná odpověď, ale doba běhu je náhodná
- Náhodnost zde užita k nalezení kratší cesty ke správné odpovědi
- Někdy selže – vybraná cesta nevede k řešení, pak nový start alg., až platné řešení
- Složitost lze chápat jako náh. prom.

- LV lze změnit na MC tím, že vydá libovol. odpověď, když výpočet nestihne v určeném čase

27

Las Vegas

Př.: randomizovaný quicksort
- není nutný medián
- jednodušší
- nezávislost na distribuci vstup. dat

28

Př. Randomizovaný quicksort
Vstup: $a_1,..,a_n$, a_i je z množiny S

function RQS (S)
if n = 1 **then** output (S);
else begin
Vyber i od 1 do n rovnoměrně náhodně
(stejná pravděp. výběru každého indexu);
Vytvoř $S_<$, $S_=$, $S_>$;
$S_<$:= [b z S: b<a_i]
$S_=$:= [b z S: b=a_i]
$S_>$:= [b z S: b>a_i]
Rekurzivně seřaď $S_<$, $S_>$ **end**
Výstup: RQS($S_<$), $S_=$, RQS($S_>$)

29

Randomizovaný quicksort

- Poskytuje správný výstup
- $\theta(n \log n)$ čas na lib. výstup s velkou pravděpodobností
- Pokud máme smůlu a vždy náhodně vybere nejmenší nebo největší prvek, je z toho Selection Sort a běží v $O(n^2)$, ale je to velmi nepravděpodobné

30

Př. k-tý nejmen. prvek z množiny (percentile, quantile)

Vstup: S = $\{a_1,..,a_n\}$ - množina prvků, a přirozené číslo $k \leq n$

Random_Select (S,k)

Step 1: **if** n = 1 **then** return a1
else vyber i od 1 do n rovnoměrně náhodně

Step 2: $S_< := \{b \text{ z } S: b<a_i\}$
$S_> := \{b \text{ z } S: b>a_i\}$

Step 3: **if** $|S_<| > k-1$ **then** Random_Select ($S_<$,k)
else if $|S_<| = k-1$ **then** return a_i
else Random_Select ($S_>$,$k-|S_<|-1$)

obecně $|S_=|$

Výstup: k-tý nejmenší prvek S
(tj. a_i: $|\{b \text{ z } S: b<a_i\}|=k-1$)

31

Analýza tohoto algoritmu

- Nejlep. výběr: $|S_<| = k-1$, ale to je řez mediánem, ten ale hledáme ☺
- Špatný výběr: $|S<|<<$ nebo $|S<|>>$, vede na $\theta(n^2)$ - nepravděpodobné
- Rozumný výběr: $|S_<| \approx |S_>|$
- V průměru: dobrý pivot je cca 25 až 75% percentile; taková hodnota má p=0.5 a srazí velikost pole na ¾

=> $O(n)$ pro libovolný vstup

32

Monte Carlo algoritmy

- S jednostrannou chybou – pouze pro rozhodovací problémy, neřekne 'ano', pokud vstup nemá požad. vlastnost, smí někdy říci 'ne', i když vstup tuto vlastnost má – velmi praktické, někdy drastická redukce složitosti oproti determin. alg.
- S oboustrannou chybou – nechat alg. běžet t-krát a vzít převažující výsledek
- S neohraničenou chybou – obecný rand. alg., může potřebovat exponenciální počet běhů pro podstatný nárůst pravděp. úspěchu

33

3. Příklady

Praktický dotaz: Jak zajistíte náhodné pořadí zpracování prvků v poli, ale zpracování všech?

34

Příklady

Praktický dotaz: Jak zajistíte náhodné pořadí zpracování prvků v poli, ale zpracování všech?

Naplnit pomocné pole všemi indexy
„Hodněkrát" prohodit dvojice prvků tohoto pole

35

Příklady

Př.: Opět zloděj s batohem omezené kapacity, tentokrát nesmí otvírat krabice a už vůbec ne lámat kusy pečiva (viz cvičení)

- Zde greedy algoritmus už neposkytne přesné optimum
- Pro celočíselné velikosti položek a kapacitu batohu (tzv. integer knapsack) lze řešit přesně a efektivně dynamickým programováním (viz později)
- Pokud kapacita moc velká, lze řešit celočíselným programováním nebo backtrackingem (viz později)
- Obecná formulace bez různých omezení: NP-úplný, ale „snadný" těžký problém

36

Příklady

Př.: Vylepšení řešení integer knapsack problému randomizací

Napřed greedy algoritmus pro nalezení počát. řešení, pak randomiz. alg. pro jeho vylepšení

Náhodně vybranou položku odstraníme z batohu, nahradíme jinými položkami (lze je vybrat stejným greedy alg.), pokud je řešení lepší, vezmeme je, pokud ne, necháme staré

Opakujeme předchozí krok podle potřeby (např. až v *k* pokusech nenastane zlepšení)

Patrně nedostaneme optimum, ale můžeme vylepšit původní řešení

37

Příklady

Př.: Hra s nulovým součtem pro 2 osoby

Red vybere řádku a zároveň Blue sloupec

Odpovídající prvek v matici je suma, kterou R dostane od B

Opakuje se podle libosti

Blue

Red				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

Nejlepší strategie: hrát nepredikovatelně, tj. náhodně, to ale neznamená „rovné pravděpodobnosti", např. B asi nezvolí 0-tý sloupec příliš často

38

Randomiz. alg. pro aproximaci strategie

Hra

Počát. předpokl.: každý hráč každý řádek/sloupec vybral 1x, (1:1:1:1)

Podle tohoto skóre náhodně vybereme tah pro R, např. řádek 0 (25,3,-14, -8)

Nejlepší tah pro B by byl sloupec 2 (-14), takže změníme skóre B na (1:1:2:1)

Na základě tohoto skóre vybereme tah pro B (např. sloupec 1: (3,12,-8,0))

Blue

Red

25	3	-14	-8
-9	12	-6	5
7	-8	-16	7
3	0	8	-2

39

Hra

Nejlepší tah pro R by byl ř.1 (12), takže změníme R skóre na (1:2:1:1)

Pokračujeme několik set tahů

Blue

Red

25	3	-14	-8
-9	12	-6	5
7	-8	-16	7
3	0	8	-2

40

Hra

Příklad náhodné hry:

Start s 2x (1:1:1:1)

R náhodně vybere 0, takže B vybere 2

Zatím je optimální strategie:

R (1:1:1:1), B(1:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

41

Hra

B náhodně vybere 1, takže R vybere 1

Zatím je optimální strategie:

R (1:2:1:1), B(1:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

42

Hra

R náhodně vybere 1, takže B vybere 0

Zatím je optimální strategie:
R (1:2:1:1), B(2:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

43

Hra

B náhodně vybere 1, takže R vybere 1

Zatím je optimální strategie:
R (1:3:1:1), B(2:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

44

Hra

R náhodně vybere 1, takže B vybere 0

Zatím je optimální strategie:
R (1:3:1:1), B(3:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

45

Hra

B náhodně vybere 2, takže R vybere 3

Zatím je optimální strategie:
R (1:3:1:2), B(3:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

46

Hra

Shrnutí náhodné hry:

R0, B2, B1, R1, R1, B0, B1, R1, R1, B0, B2, R3

vedlo na zatím optimální strategii

R (1:3:1:2), B(3:1:2:1)

Red \ Blue				
	25	3	-14	-8
	-9	12	-6	5
	7	-8	-16	7
	3	0	8	-2

47

4. Náhodná čísla

- Někdy potřebujeme pro ně samotná (míchání v karetní hře), někdy v simulaci (fronty), pro vyhnutí se obtížnému matem. problému, pro náhodné vzorkování nebo pro randomiz. algoritmus
- Na determin. zařízení nelze dosáhnout úplné náhodnosti, můžeme doufat jen v pseudonáh. čísla
- Špatný generátor – vážné důsledky (viz nedávné problémy s bezpečností internetového schématu hesel – krátká perioda, takže brute-force hledání rychle vyčerpalo všechny kombinace)

48

- Nejčastěji lineární kongruentní generátor
  $R_n = (aR_{n-1} + c) \bmod n$
  např. pro 32b $R_0 = 0$, a=1366, c=150 889,
  m=714 025 apod.
- Hodnoty bez znalosti teorie neměnit („domácí" vylepšení obvykle zhorší)
- Jiná neunif. rozložení: metoda přijetí-odmítnutí
  (ohraničíme požad. distrib. fci nebo geom. oblast do boxu, pak vybereme náh. bod p z boxu nezávislou generací x,y souřadnice, bod přijmeme, pokud je uvnitř – správné, ale pomalé)

49

5. Závěr

- Randomizované algoritmy jsou často experimentální – skoro jistě nedostaneme perfektní nebo optimální výsledky, ale „hezky dobré" výsledky
- Randomizovaný algoritmus je to, co použijeme, když nevíme, co dělat jiného – měl by být v toolboxu každého programátora!

50