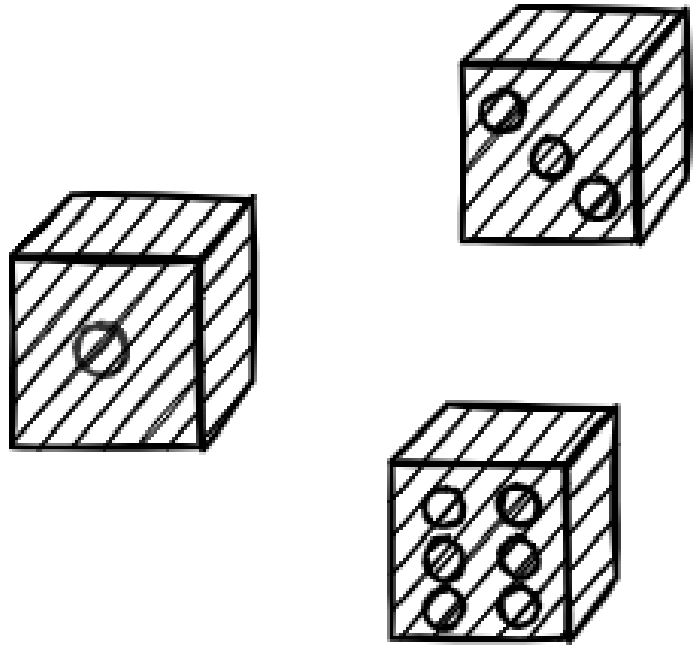
Naključnostni algoritmi

Uroš Čibej



Pregled snovi

1. Naključnost v algoritmiki
2. Primeri
    i. quicksort
    ii. največji prerez
    iii. 3SAT
    iv. PZT
    v. ujemanje v dvodelnih grafih

Literatura

https://introtcs.org/public/lec_16_randomized_alg.html

Naključnost v algoritmiki

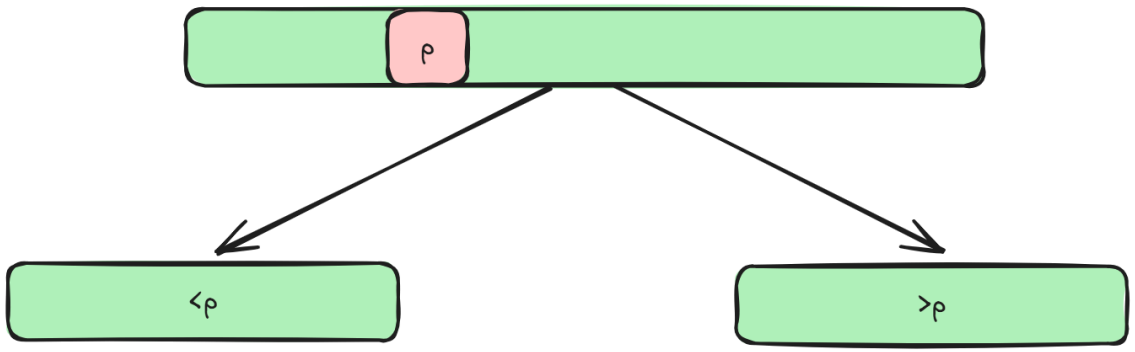
- poenostavitev algoritmov
- pohitritev algoritmov
- razbijanje vzorcev (nepredvidljivost)
- odpornost na napade
- boljši izračuni spodnjih (zgornjih) mej
- razbijanje simetrij

Zgodovina

- 1940 (projekt Manhattan) metode poimenovane Monte Carlo
- 1970 (Rabin-Miller) testiranje praštevilskosti
- 1980-90 razredi računske zahtevnosti (BPP, ZPP, RP)
- 1990-00 presenetljive meje za aproksimacijo
- 2010- ali je naključnost sploh potrebna

Qsort

- Tony Hoare
- Deterministične izbire pivota
- Analiza povprečne zahtevnosti



Analiza pričakovanega časa izvajanja

Opazke:

1. Pri analizi porabe časa štejemo št. primerjav
2. Zanima nas pričakovano število primerjav
3. Dva elementa med sabo primerjamo največ 1x (ko je en izmed njiju pivot)
4. Kolikšna je verjetnost, da bomo dva elementa a_i in a_j med sabo primerjali?

Imejmo indikatorsko spremenljivko C_{ij}, ki pove ali se je zgodila primerjava med elementoma a_i in a_j

Verjetnost primerjave

1. Primerjava se zgodi, če je a_i ali a_j izbran za pivot
2. Če je za pivot izbran a' ($a_i < a' < a_j$), ta dva elementa nikoli ne bosta primerjana
3. Kritični trenutek je torej, ali na intervalu $[i, \ldots, j]$ izberemo bodisi i ali j ali en element vmes

$$P[a_i \text{ primerjamo z } a_j] = \frac{2}{j - i + 1}$$

Skupno število primerjav

$$\sum_i \sum_{j>i} C_{ij}$$

Pričakovana vrednost te količine:

$$E\left(\sum_i \sum_{j>i} C_{ij}\right) = \sum_i \sum_{j>i} E(C_{ij}) = \sum_i \sum_{j>i} \frac{2}{j-i+1}$$

Največji prerez

$$G = \langle V, E \rangle$$

S_1, S_2 razbitje V.

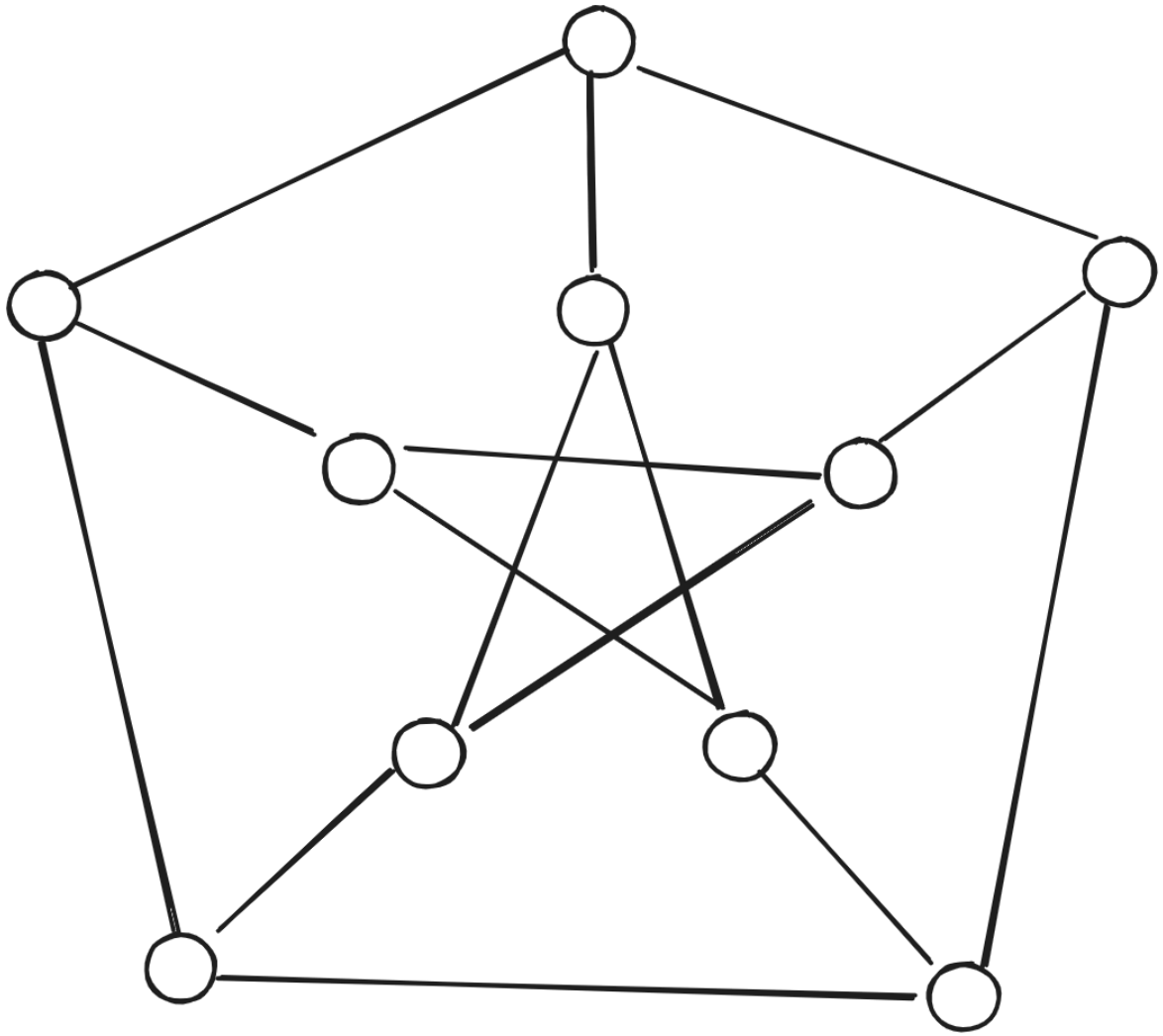
Prerez:

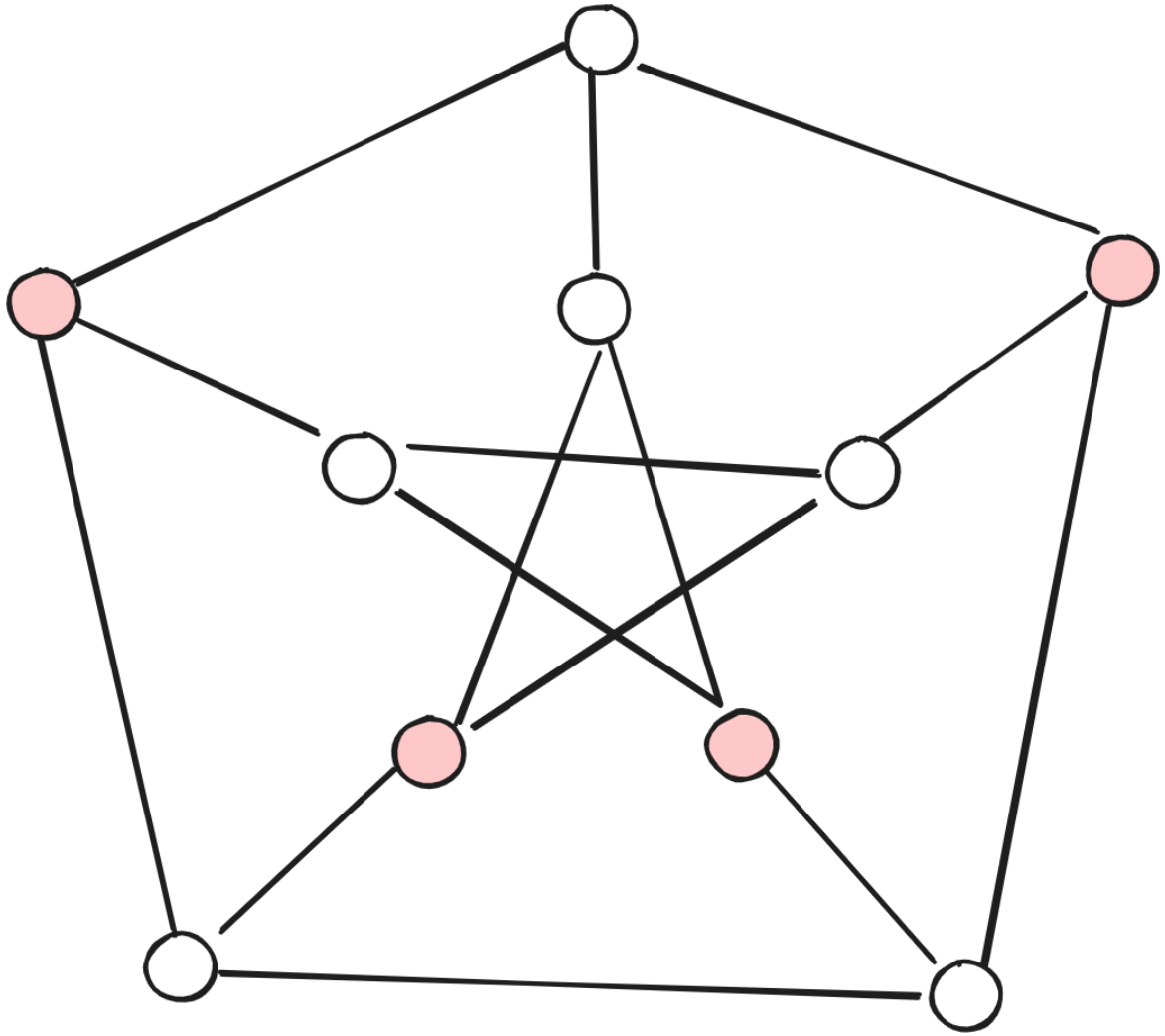
$$C = \{(u, v) \mid u \in S_1, v \in S_2\}$$

Maks:

$$|C|$$

Primer





Naključnostni algoritem

Vsako vozlišče naključno damo v S_1 ali S_2

Analiza upanja

C_{ij} indikatorska spremenljivka - povezava je v prerezu

$P[(i,j) \text{ je v prerezu}] = \frac{1}{2}$

$$E\left[\sum_{(i,j)\in E} C_{ij}\right] = \frac{m}{2}$$

Verjetnost takega izida

Izrek: Verjetnost, da ima naključen prerez vsaj $\frac{m}{2}$ povezav je $\geq \frac{1}{2m}$

Ideja dokaza

Pokazati moramo, da verjetnost ne more biti majhna, če je upanje (povprečje) $\frac{m}{2}$

Primer

$m = 1000$ povezav $\implies$ povprečje je 500 povezav

recimo, da se zgodi "najslabši" scenarij

- 99.9% 499 povezav
- 0.1% 1000 povezav

Povprečje:

$$0.999 \cdot 499 + 0.001 \cdot 1000 = 499.501 < 500$$

Verjetnost, da izberemo prerez ≥ 500 mora torej biti večji od 0.1

Dokaz

Najslabši scenarij na splošno:

- $(1 - P)$ prerez $\frac{m}{2} - 1$
- P prerez m

$$P \cdot m + (1 - P)\frac{m}{2} - 1$$

Koliko je lahko P najmanj, da je upanje še vedno $m/2$?

$P \geq \frac{1}{2m}$

Ojačanje rezultata

Izrek: Obstaja algoritem, z vhodom $G = \langle V, E \rangle$ in $k > 0$, polinomski v odvisnosti od n, k, ki vrača prerez (S_1, S_2), $|$(S_1,S_2)$| \geq \frac{m}{2}$ z verjetnostjo $\geq 1 - 2^{-k}$

```
def large_cut(G, k):
  for _ in 1..200*k*m:
    x = naključen vektor iz {0,1}^n
    (S1,S2) = get_cut(G, x)
    if |S1, S2|>= m/2:
        return S1,S2
  return None
```

Naključnost vs. Determinističnost

- Fizikalni argument: vsak program je že naključen zaradi zunanjih dejavnikov
- Algoritmi, kjer verjetnost napake pada eksponentno, so ekvivalentni deterističnim

Dve vrsti naključnostnih algoritmov

- **Monte Carlo**
  - čas izvajanja je klasičen, izid ni optimalen
- **Las Vega**s
  - izid je pravilen, optimalen, zanesljiv, čas izvajanja odvisen od naključja

3SAT z naključjem

Groba ideja:

1. izberemo naključno dodelitev vrednosti spremenljivkam
2. lokalno izboljšujemo
    i. izberemo naključno klavzulo z vrednostjo 0
    ii. v tej klavzuli naključnemu literalu spremenimo vrednost

Walk3SAT

```
def walk3SAT(phi):
  for i in 1..S:
    x = naključen dvojiški vektor dolžine n
    for j in 1..T:
      if phi(x) == 1:
        return x
      unsat_clauses = [c for c in phi and c[x] = 0]
      l1,l2,l3 = naključna klavzula iz unsat_clauses
      l = naključni literal izmed l1,l2,l3
      x[l] = not x[l]
  return "UNSAT"
```

Analiza

- Čas izvajanja je $O(S \cdot T \cdot poly(n))$
- Kako določiti S in T, da bo verjetnost napake dovolj majhna?

Oznake:

- z x bomo označili neko trenutno dodelitev vrednosti
- z x^* bomo označili neko dodelitev pri kateri $\phi[x] = 1$
- $\Delta(x, y)$ je Hammingova razdalja med dvema dodelitvama (binarna vektorja dolžine n)

Verjetnost lokalne izboljšave

Izrek: Verjetnost, da se razdalja zmanjša za 1 v enem koraku je $\frac{1}{3}$

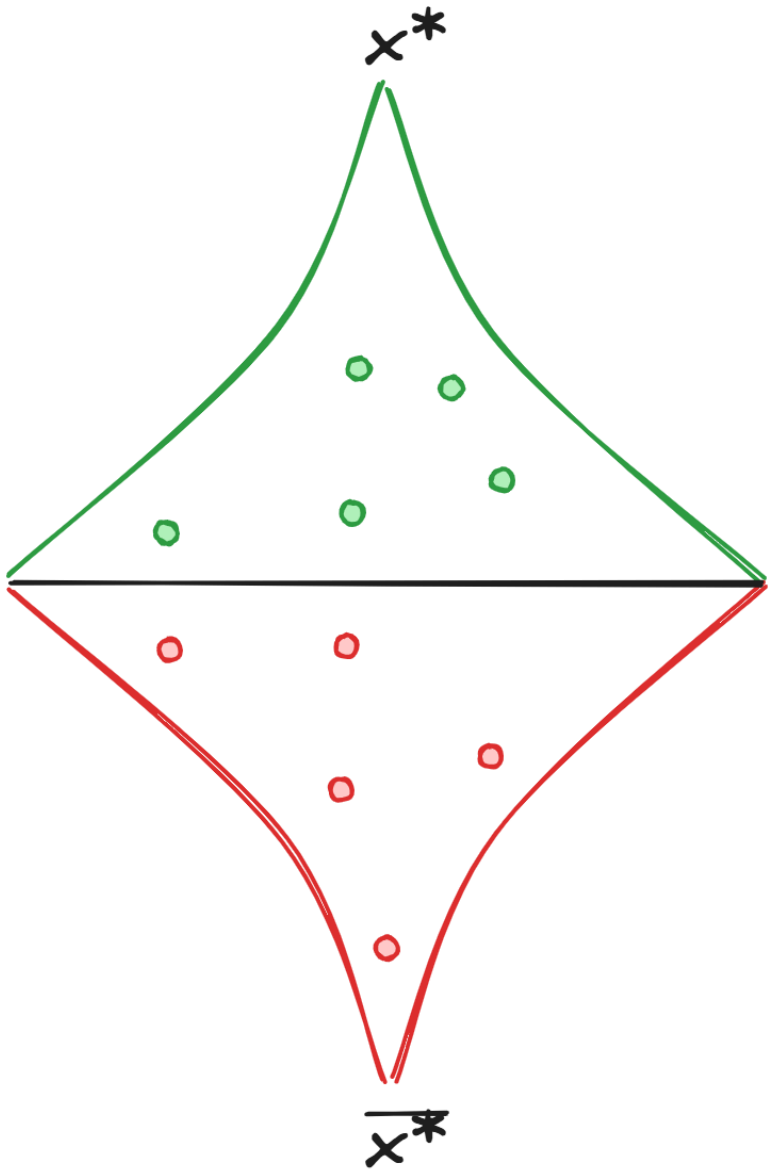
Dokaz Ko izberemo klavzulo (l_1, l_2, l_3), ima vsaj ena spr. v tej klavzuli drugačno vrednost v x kot v x^*. Z verjetnostjo $\frac{1}{3}$ izberemo točno to spremenljivko in $\Delta(x, x^*)$ zmanjšamo za 1.

Začetna $\Delta(x, x^*)$

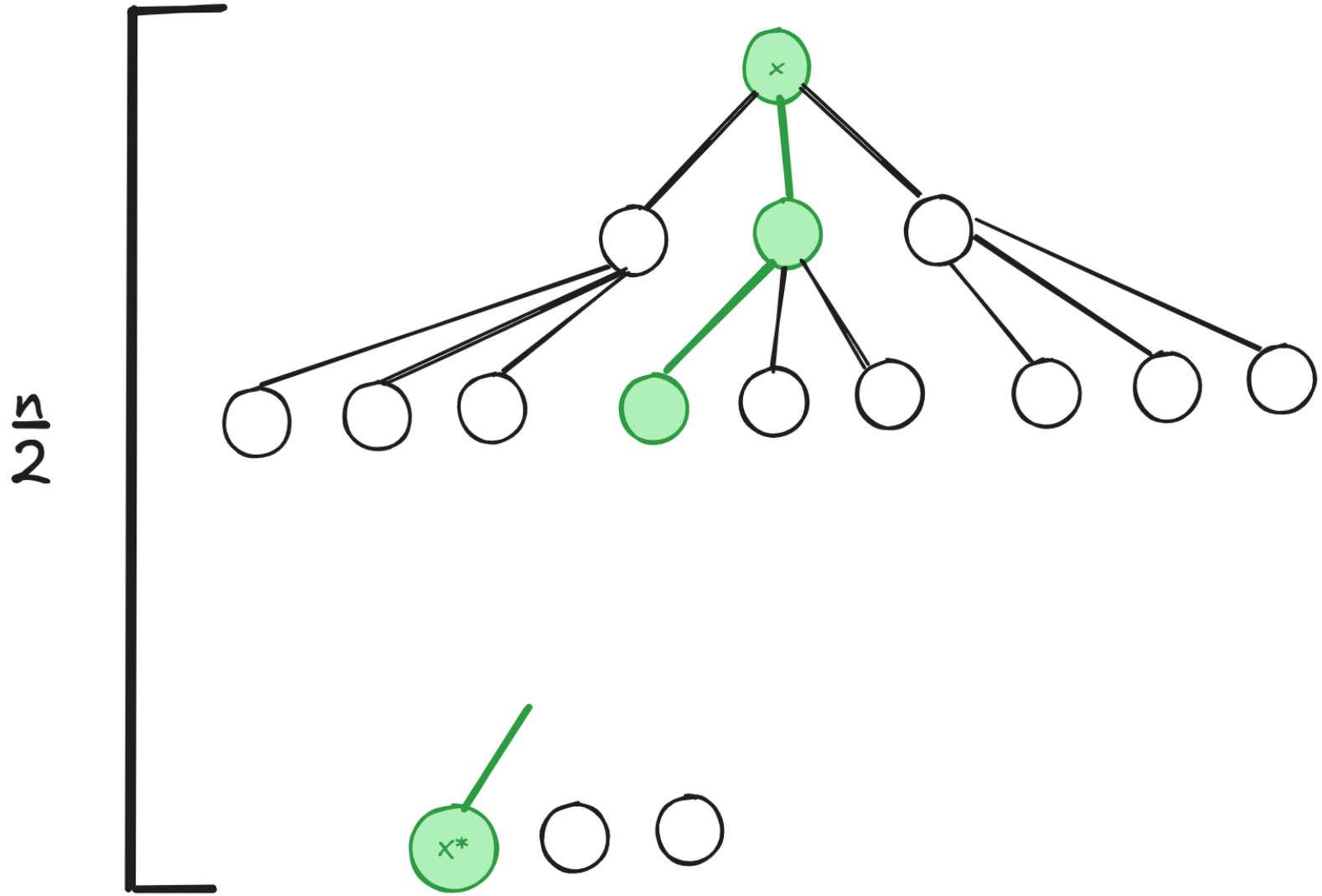
Izrek: Verjetnost, da je $\Delta(x, x^*) \leq \frac{n}{2}$ pri naključnem x je $\frac{1}{2}$

- vsak vektor $y \in \{0,1\}^n$ ima "dvojčka" $\overline{y}$.
- $\Delta(x^*, \overline{x^*}) = n$
- $\forall y \in \{0,1\}^n$:

  $\Delta(y, x^*) \leq \frac{n}{2} \vee \Delta(\overline{y}, x^*) \leq \frac{n}{2}$



Ko izberemo "pravo" polovico:



Zaključek (Walk3SAT)

Izrek Naj bo $T = 100 \cdot \sqrt{3}^n$ in $S = \frac{n}{2}$.
Tedaj je verjetnost, da algoritem za zadovoljivo formulo φ vrne $UNSAT$, največ $\frac{1}{2}$.

Dokaz

1. V eni iteraciji zunanje zanke uspemo najti x^* vsaj z verjetnostjo $\frac{1}{2\sqrt{3}^n}$
2. Verjetnost, da v nobeni od $T = 100 \cdot \sqrt{3}^n$ ponovitev ne uspemo najti je:

$$\left(1 - \frac{1}{2\sqrt{3}^n}\right)^{100 \cdot \sqrt{3}^n} \leq \frac{1}{e}^{50} \leq \frac{1}{2}^{50}$$

pomožna neenakost
$((1 - \frac{1}{k})^k \leq \frac{1}{e} \leq \frac{1}{2})$

Testiranje ničelnosti polinoma (PZT)

Problem:

$$p(x_1, x_2, \ldots, x_n) = 0$$

$x_i \in F$

Denimo, da imamo polinom kot "črno škatlo", zanima nas, če je enak ničelnemu polinomu

Ena spremenljivka

deterministični algoritem: O(d)

1. Polinom stopnje d ima največ d ničel.
2. Če izberemo $n+1$ različnih vrednosti $v_1, \ldots, v_{d+1}$ in je $p(v_i) = 0$ pri vseh vrednostih, potem $p \equiv 0$

Ena spremenljivka

naključnostni algoritem: O(1)
$S \subseteq F, |S| \geq 2d$

1. izberemo naključno vrednost $v \in S$
2. če $p(v) = 0$ vrnemo $p \equiv 0$
3. sicer $p \not\equiv 0$

Delež ničel polinoma je največ $\frac{d}{|S|} \leq \frac{1}{2}$

- Če $p \equiv 0$ vedno dobimo vrednost 0 zato je izhod $p \equiv 0$
- Če $p \not\equiv 0$ vrnemo $p \not\equiv 0$ z verjetnostjo $\geq \frac{1}{2}$

Več spremenljivk

Izrek (Schwartz-Zippel): Naj bo $q \in \mathbb{N}$ in naj bo $p : \mathbb{R}^n \to \mathbb{R}$ polinom s celoštevilskimi koeficienti stopnje $\deg(p) \leq d$. Če polinom p ni enak nič, potem ima največ dq^{n-1} ničel v množici:

$$[q]^n = \{(x_0, \ldots, x_{n-1}) : x_i \in \{0, \ldots, q-1\}\}$$

Posledica:

Če imamo neničelni polinom $p(x_1, x_2, \ldots, x_n)$ stopnje d in vrednost vsake spr. izberemo naključno iz množice S.

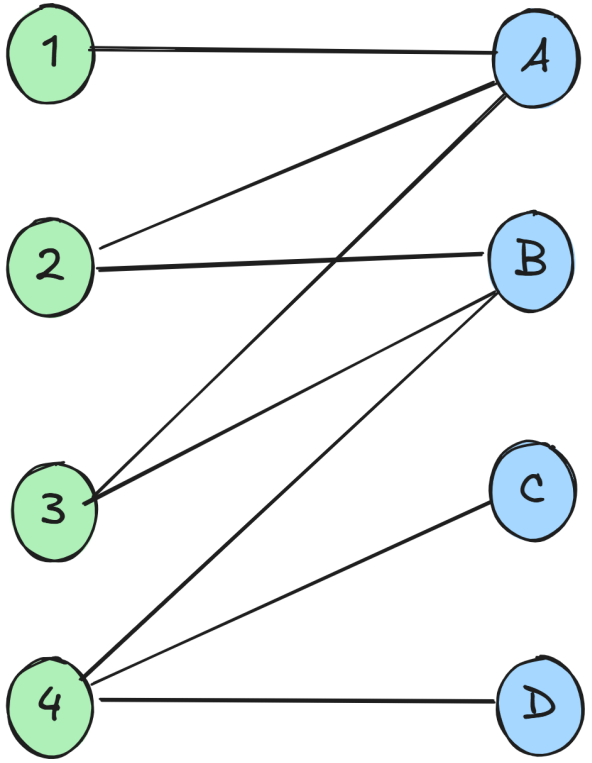
$$P[\text{ polinom se ovrednoti v } 0] \leq \frac{d}{|S|}$$

Popolno ujemanje

Problem PM :

vhod: dvodelni graph $G = \langle U, V, E \rangle$

vprašanje: Ali v G obstaja popolno ujemanje?



Determinanta Tutte-jeve matrike

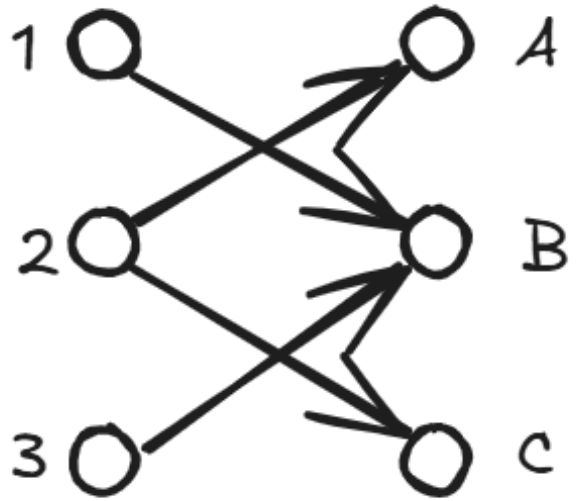
Leibnizova definicija determinante

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Primer 3×3

$$\det(A) = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}.$$

$$A = \begin{pmatrix} 0 & a_{1B} & 0 \\ a_{2A} & 0 & a_{2C} \\ 0 & a_{3B} & 0 \end{pmatrix}$$



$$\det(A) = 0 \cdot 0 \cdot 0 + a_{1B} \cdot a_{2C} \cdot 0 + 0 \cdot a_{2A} \cdot a_{3B} - 0 \cdot 0 \cdot 0 - 0 \cdot a_{2C} \cdot a_{3B} - a_{1B} \cdot a_{2A} \cdot 0 = 0.$$

Algoritem

```
def has_perfect_matching(G=<U,V, E>):
  A = matrika ničel dimenzije nxn
  for i,j in UxV:
    if (i,j) ni povezava v grafu:
        A[i,j] = 0
    else:
        A[i,j] = naključno število na intervalu [0, ... , 2n-1 ]

    if det(A) == 0:
      return False
    else:
      return True
```

Kako s tem algoritmom dobimo dejanski prerez?