Algoritmi Probabiliști

Algoritmi Probabiliști

- În timpul rezolvării unei probleme, putem ajunge la un moment dat în situaţia de a avea **de ales** între mai multe variante de continuare.

Algoritmi Probabiliști

- În timpul rezolvării unei probleme, putem ajunge la un moment dat în situaţia de a avea **de ales** între mai multe variante de continuare.
    - **se alege aleator una dintre variante**

Algoritmi Probabiliști

- În timpul rezolvării unei probleme, putem ajunge la un moment dat în situaţia de a avea **de ales** între mai multe variante de continuare.
  - **se alege aleator una dintre variante**
  - **la executări diferite ale unui algoritm probabilist, rezultatele sunt în general diferite.**

Algoritmi Probabiliști

Categorii

- Monte Carlo
- Las Vegas
- Algoritmi numerici

Algoritmi Monte Carlo

Algoritmi Monte Carlo

- Furnizează totdeauna un rezultat, care însă **nu este neapărat corect**
- **Probabilitatea ca rezultatul să fie corect crește pe măsură ce timpul disponibil crește**

Algoritmi Monte Carlo

Se consideră un vector cu *n* elemente distincte. Să se determine un element al vectorului care să fie mai mare sau egal cu mediana a celor n numere din vector

- n este foarte mare
- timpul avut la dispoziţie este mic

Algoritmi Monte Carlo – Mediana

$v = -\infty$

Repetă fără a depăşi timpul disponibil:

- alegem aleatoriu x un element al vectorului
- v = maxim(v, x) = cel mai mare element ales

scrie v

Algoritmi Monte Carlo – Mediana

- Care este probabilitatea ca un element ales x să fie mai mic/mai mare decât mediana?

- **Care este probabilitatea ca răspunsul să fie corect/greșit după k încercări?**

Algoritmi Monte Carlo – Mediana

- Care este probabilitatea ca un element ales x să fie mai mic/mai mare decât mediana?
  - probabilitatea să fie mai mare sau egal $\geq 1/2$
  - probabilitatea să fie mai mic $\leq 1/2$

- **Care este probabilitatea ca răspunsul să fie corect/greșit după k încercări?**

Algoritmi Monte Carlo – Mediana

- **Care este probabilitatea ca răspunsul să fie greșit după k încercări?**

    - Probabilitatea ca **toate** cele k elemente alese (în timpul de rulare avut la dispoziţie) să fie mai mici decât mediana (deci v < mediana) $\leq \left(\frac{1}{2}\right)^k$

Algoritmi Monte Carlo – Mediana

- Care este probabilitatea ca răspunsul să fie corect după k încercări?

  $1 - 1/2^k$

  - **Pentru k=20, această probabilitate este mai mare decât 0,999999**

Algoritmi Monte Carlo – Mediana

- Care este probabilitatea ca răspunsul să fie corect după k încercări?

  $1 - 1/2^k$

  - **Pentru k=20, această probabilitate este mai mare decât 0,999999**

- **$1 - (1-p)^k$ – corect**
  - $1-p$ = probabilitatea de a greși la un pas

Algoritmi Monte Carlo

Se consideră un vector cu *n* elemente. Să se determine **dacă există** un element majoritar în vector (cu frecvența > n/2)

- Mai general – cu frecvența >fr·n

Algoritmi Monte Carlo – Element majoritar

$v = -\infty$

Repetă fără a depăşi timpul disponibil:

- alegem aleator x un element al vectorului
- Calculam f = frecventa lui x
- Daca f>n/2 scrie DA; STOP

scrie NU

Algoritmi Monte Carlo – Element majoritar

Analiza

- **Problemă de decizie**
- Dacă scrie **DA, răspunsul este corect**
- Dacă scrie NU, este posibil ca să existe element majoritar (deci răspunsul să fie greșit)

Algoritmi Monte Carlo – Element majoritar

Analiza

- **Problemă de decizie**
- Dacă scrie DA, răspunsul este corect
- Dacă scrie NU, este posibil ca să existe element majoritar (deci răspunsul să fie greșit)
- Care este probabilitatea de a **răspunde greșit NU** după k încercări?

Algoritmi Monte Carlo – Element majoritar

Analiza

- **Problemă de decizie**
- Dacă scrie DA, răspunsul este corect
- Dacă scrie NU, este posibil ca să existe element majoritar (deci răspunsul să fie greșit)
- Care este probabilitatea de a **răspunde greșit NU** după k încercări?
  - Dacă există element majoritar, probabilitatea să nu îl găsească după k încercări este $\leq 1/2^k$

Algoritmi Monte Carlo

Algoritmul lui KARGER de determinare a unei tăieturi minime într-un graf

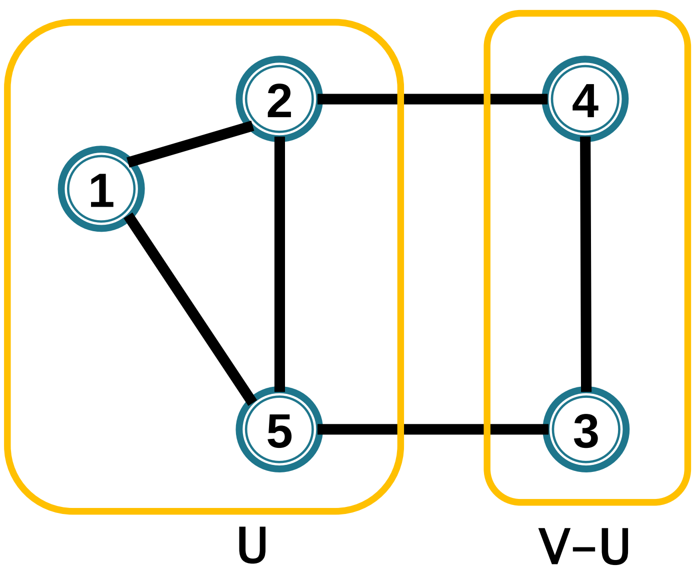
- Jon Kleinberg and Éva Tardos - Algorithm Design
- D. R. Karger, Global min-cuts in rnc, and other ramifications of a simple min-out algorithm. In Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 1993
- D. R. Karger, S. Clifford, A new approach to the minimum cut problem, Journal of the ACM. 43 (4), 1996 doi:10.1145/234533.234534.
- https://en.wikipedia.org/wiki/Karger's_algorithm

Algoritmul lui KARGER

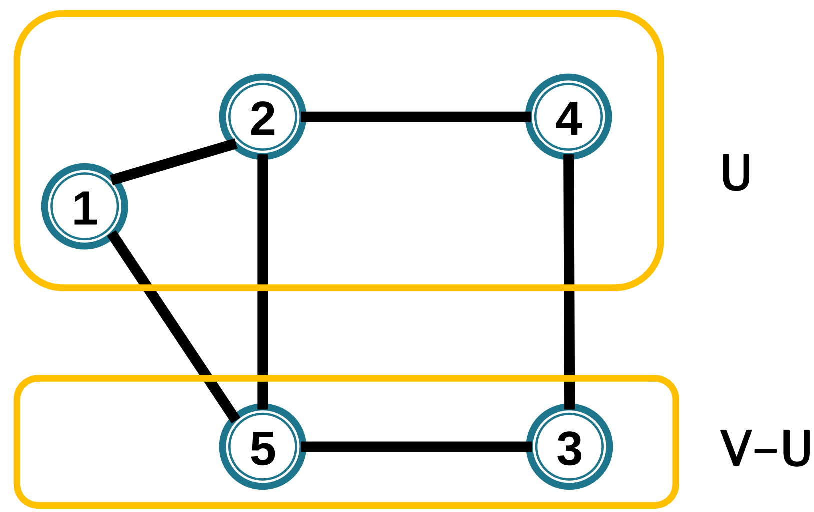
- **Tăietură într-un graf neorientat G=(V,E)**
  - = partiționare a mulțimii vârfurilor (U,V−U)
  - – muchiile de la U la V−U sunt muchiile tăieturii

Cardinalul tăieturii = numărul de muchii ale tăieturii

Tăietură minimă = de cardinal minim



cardinal 2



cardinal 3

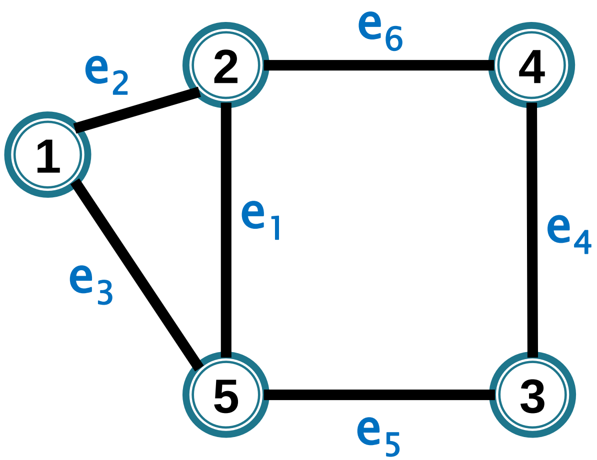
Algoritmul lui KARGER

- **Tăietură minimă într-un graf neorientat G=(V,E)**
  - **Soluție – bazată pe fluxuri maxime în rețele de transport**
    - cu ajutorul fluxurilor putem determina s-t tăietură minimă într-un graf **orientat**, pentru s, t fixate (o s-t tăietură= tăietură (U,V-U) în care $s \in U$, $t \notin U$) – **Soluție**: transformăm graful în graf orientat cu toate capacitățile arcelor 1, repetam algoritmul pentru $t \in V$ (v. Jon Kleinberg and Éva Tardos – Algorithm Design)
  - **Aplicații – probleme de conectivitate**

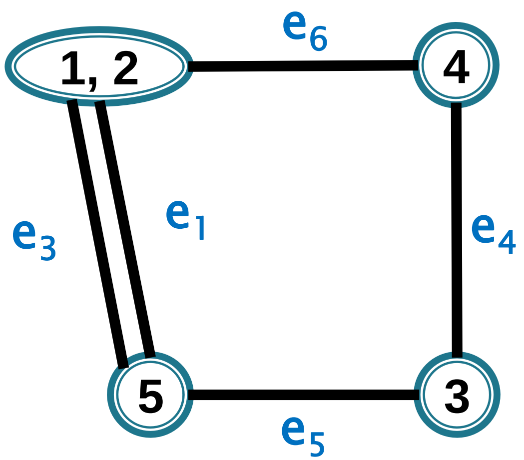
Algoritmul lui KARGER

- **Idee**
  - Se alege aleatoriu o muchie și se contractă (eliminând buclele) până când multigraful obținut are două vârfuri u și v
  - U = mulțimea vârfurilor contractate în u

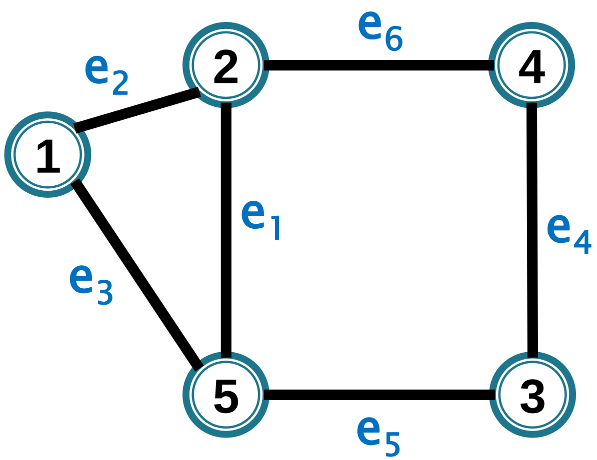
Algoritmul lui KARGER



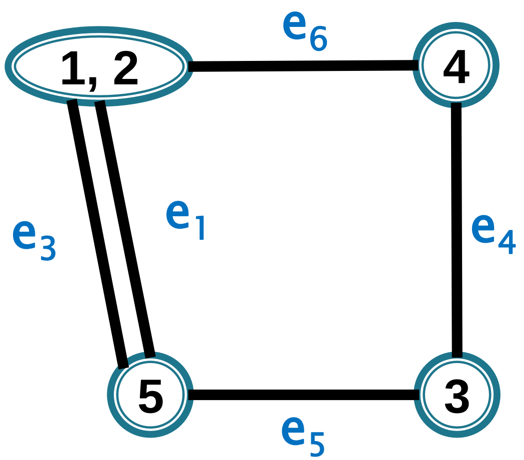
contracta e_2



Algoritmul lui KARGER

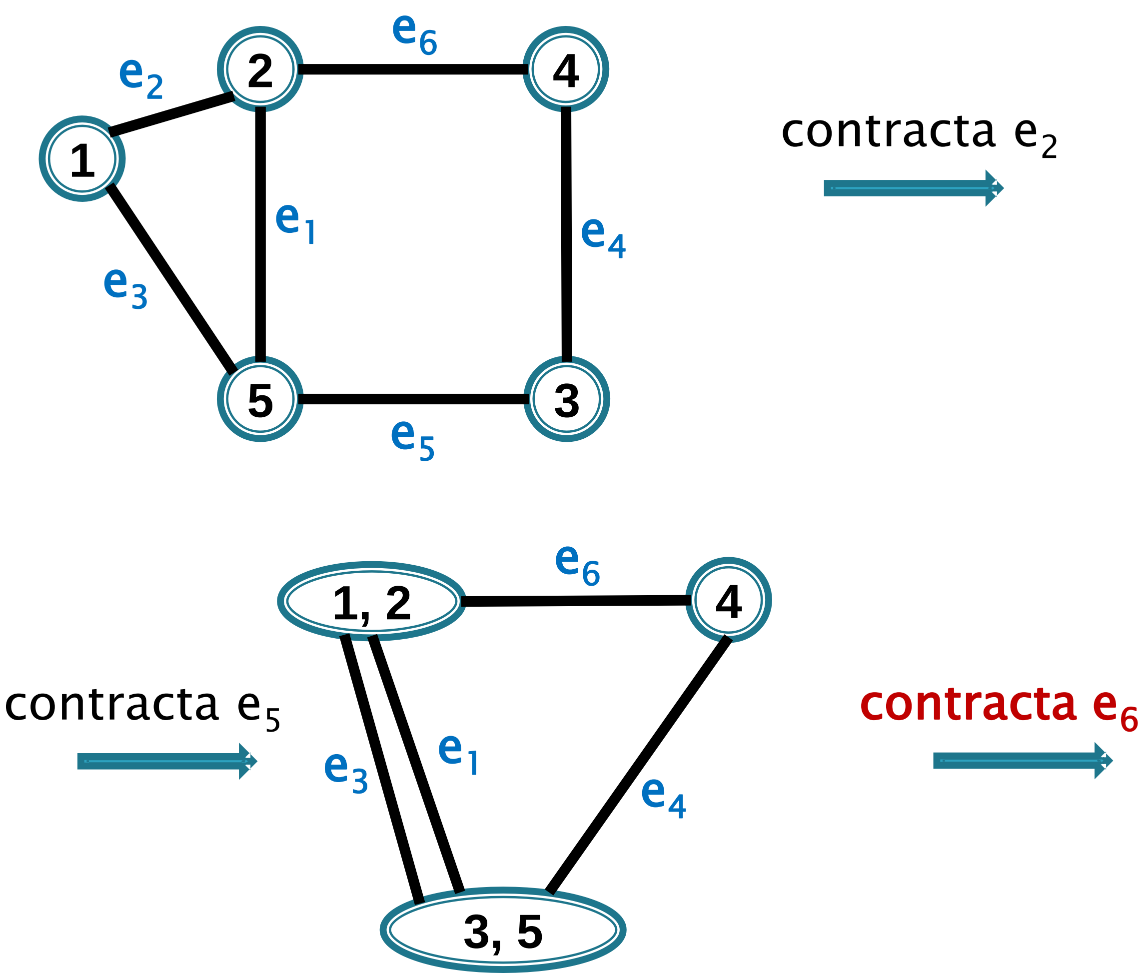


contracta e_2

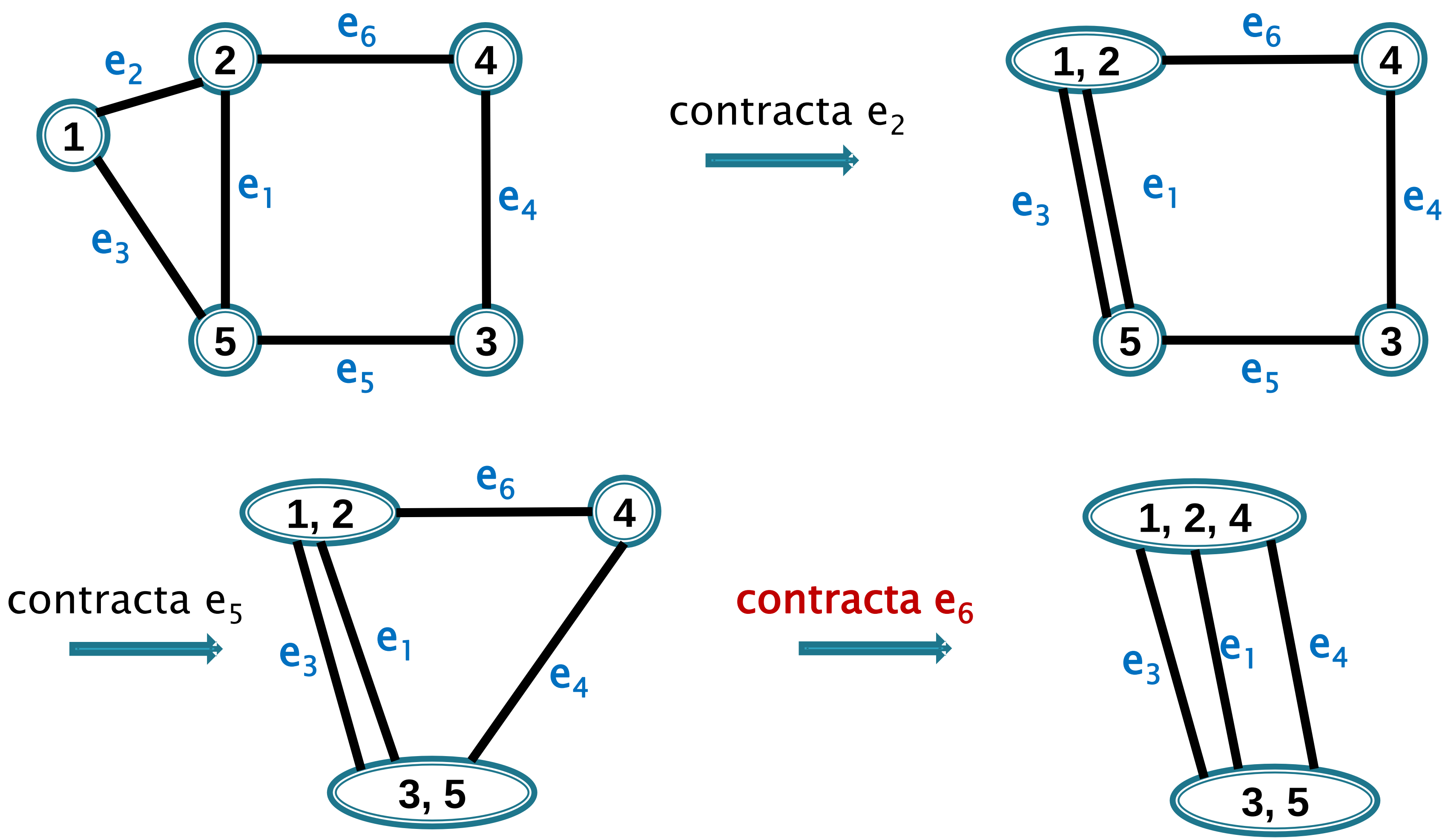


contracta e_5

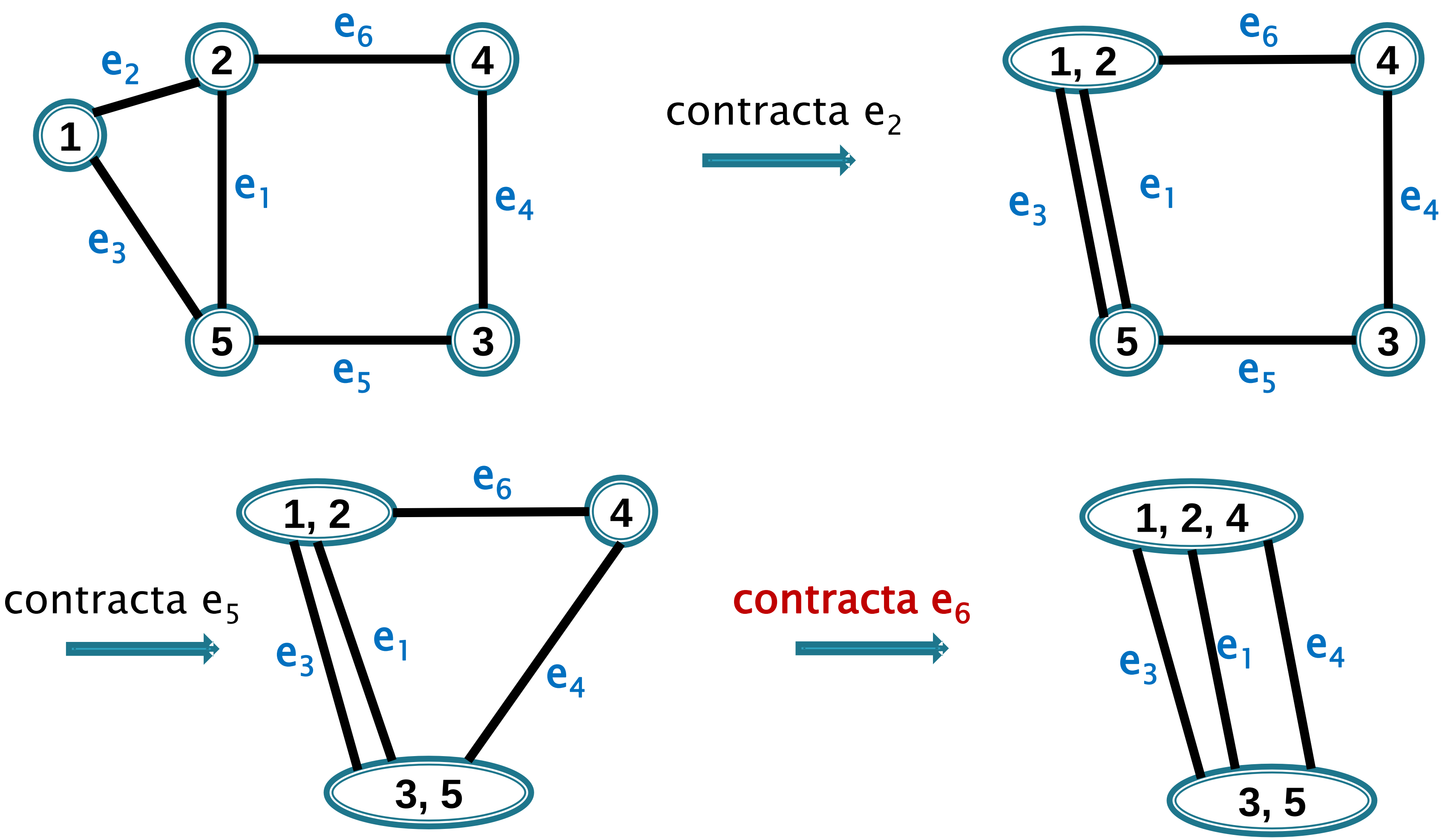
Algoritmul lui KARGER



Algoritmul lui KARGER

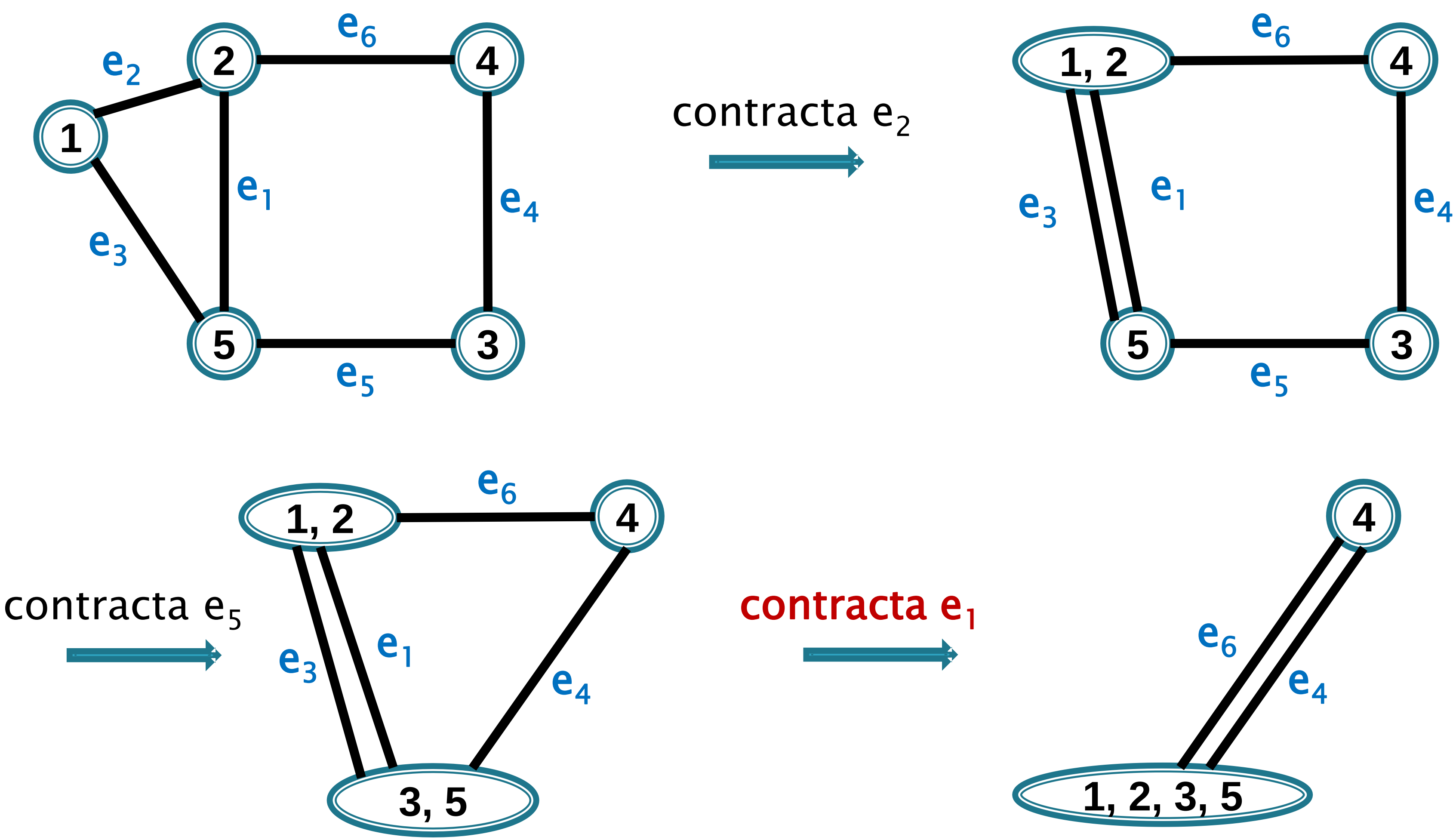


Algoritmul lui KARGER



Tăietura obținută are cardinal 3, algoritmul nu furnizează mereu o soluție optimă

Algoritmul lui KARGER



Algoritmul lui KARGER

- **Pseudocod – pentru o etapă** (se repetă de un număr de ori)

pentru v ∈ V executa **M(v) ← {v}**

cat timp |V(G)| >2 executa

    alege aleator o muchie e=uv

    contracta muchia e obtinand un supernod n_{uv}

    $M(n_{uv}) \leftarrow M(u) \cup M(v)$

fie u,v cele doua varfuri ramase in V(G)

U ← M(u)

returneaza (U, V-U) – formata cu muchiile din E(G)

Algoritmul lui KARGER

Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

Algoritmul lui KARGER

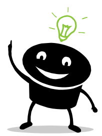
Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

- fixăm T=(U,V-U) o tăietură minimă cu mulțimea muchiilor F
- Calculăm probabilitatea ca algoritmul să returneze T

Algoritmul lui KARGER

Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

- fixăm T=(U,V-U) o tăietură minimă cu mulțimea muchiilor F
- Calculăm probabilitatea ca algoritmul să returneze T



returnează T ⇔ nu este contractată nicio muchie a tăieturii T (= nicio muchie din F)

Algoritmul lui KARGER

Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

- fixăm T=(U,V−U) o tăietură minimă cu mulțimea muchiilor F
- Calculăm probabilitatea ca algoritmul să returneze T

returnează T ⇔ nu este contractată nicio muchie a tăieturii T (= nicio muchie din F)

Care este probabilitatea ca la primul pas să fie contractată o muchie din F?

Algoritmul lui KARGER

Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

- fixăm T=(U,V–U) o tăietură minimă cu mulțimea muchiilor F
- Calculăm probabilitatea ca algoritmul să returneze T

returnează T ⇔ nu este contractată nicio muchie a tăieturii T (= nicio muchie din F)

Care este probabilitatea ca la primul pas să fie contractată o muchie din F?

$$\frac{|F|}{|E(G)|} = \frac{|F|}{m}$$

Algoritmul lui KARGER

Care este probabilitatea ca tăietura returnată de algoritm să fie minimă?

– fixăm T=(U,V–U) o tăietură minimă cu mulțimea muchiilor F

– Calculăm probabilitatea ca algoritmul să returneze T

returnează T ⇔ nu este contractată nicio muchie a tăieturii T (= nicio muchie din F)

Care este probabilitatea ca la primul pas să fie contractată o muchie din F?

$$\frac{|F|}{|E(G)|} = \frac{|F|}{m}$$

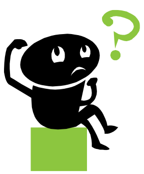
Algoritmul lui KARGER

- Probabilitatea ca la primul pas să fie aleasă o muchie din T este

$$\frac{|F|}{m}$$

- Probabilitatea ca la primul pas să **nu** fie aleasă o muchie din T este

$$1-\frac{|F|}{m}$$



$$\frac{|F|}{m} \simeq ?$$

Algoritmul lui KARGER

- Probabilitatea ca la primul pas să fie aleasă o muchie din T este

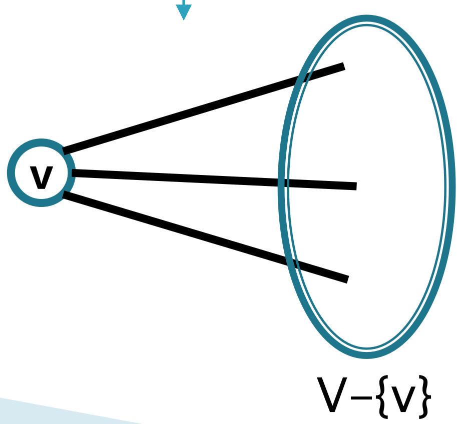
$$\frac{|F|}{m}$$

- Probabilitatea ca la primul pas să **nu** fie aleasă o muchie din T este

$$1-\frac{|F|}{m}$$

- $|F| \leq deg(v)$, $\forall v \in V$,

  deoarece ({v}, V−{v}) este tăietură de cardinal deg(v)



Algoritmul lui KARGER

- Probabilitatea ca la primul pas să fie aleasă o muchie din T este

$$\frac{|F|}{m}$$

- Probabilitatea ca la primul pas să **nu** fie aleasă o muchie din T este

$$1-\frac{|F|}{m}$$

- $|F| \leq \deg(v)$, $\forall v \in V$,

  deoarece $(\{v\}, V-\{v\})$ este tăietură de cardinal $\deg(v)$

- $$\sum_{v\in V}\deg(v) = 2m$$

Algoritmul lui KARGER

- Probabilitatea ca la primul pas să fie aleasă o muchie din T este

$$\frac{|F|}{m}$$

- Probabilitatea ca la primul pas să **nu** fie aleasă o muchie din T este

$$1-\frac{|F|}{m}$$

- $|F| \le \deg(v),\ \forall v \in V$,

  deoarece ({v}, V–{v}) este tăietură de cardinal deg(v)

- $$\sum_{v\in V}\deg(v)=2m \quad \Rightarrow \quad n\,|F|\le 2m \quad \Rightarrow \quad \frac{|F|}{m}\le\frac{2}{n}$$

Algoritmul lui KARGER

- Rezultă că probabilitatea ca la primul pas să **nu** fie aleasă o muchie din T este

$$1-\frac{|F|}{m} \geq 1-\frac{2}{n}=\frac{n-2}{n}$$

Algoritmul lui KARGER

- **După primul pas** – orice tăietură din noul graf G_1 este tăietură și în G
- După ce este contractată prima muchie, dacă aceasta nu este din F, atunci F este tăietură minimă și pentru G_1 ($|V(G_1)|=n-1$)
  - $\Rightarrow$ supernodul n_{uv} are grad $\geq |F|$

    (deoarece $(\{n_{uv}\}, V-\{n_{uv}\})$ este tăietură in G_1 de cardinal $deg(n_{uv})$)
- Raționament similar $\Rightarrow$ probabilitatea ca a doua muchie aleasă să nu fie din F, **condiționată de faptul că prima nu a fost din F** este

$$\geq 1-\frac{2}{n-1}=\frac{n-3}{n-1}$$

Algoritmul lui KARGER

- **După i pași**
  - orice tăietură din noul graf G_i este tăietură și în G
  - Similar la pasul i+1, dacă până la acel pas nu a fost contractată nicio muchie din F, probabilitatea ca muchia aleasă la acest pas să nu fie din F

$$\geq 1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}$$

Algoritmul lui KARGER

- **Formalizând**
  - Notăm A_i evenimentul: la pasul i nu este contractată o muchie din F
  - $Pr[A_{i+1} | A_1 \cap \ldots \cap A_i]$ = probabilitatea ca la pasul i+1 să nu fie contractată o muchie din F, condiționată de faptul că până la acel pas nu a fost contractată nicio muchie din F
  - Avem

$$Pr[A_{i+1} | A_1 \cap \ldots \cap A_i] \geq 1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}$$

Algoritmul lui KARGER

- Rezultă că probabilitatea ca algoritmul să returneze F (să nu aleagă la niciun pas o muchie din F) este

  $= \Pr[A_1] \cdot \Pr[A_2 | A_1] \cdot \ldots \cdot \Pr[A_{n-2} | A_1 \cap \ldots \cap A_{n-3}]$

$$\geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \ldots \cdot \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)} = \frac{1}{C_n^2} = \binom{n}{2}^{-1}$$

- $O(n^2)$ încercări – păstrăm tăietura cea mai mică obținută

- Karger – Stein – optimizare

Algoritmul lui KARGER

- Amintim relația

$$\boxed{1 - p \le e^{-p}}$$

- După k repetări, probabilitatea de eșec va fi

$$\boxed{(1 - p)^k \le e^{-pk}}$$

Algoritmul lui KARGER

$$(1-p)^k \le e^{-pk}, \quad p = \frac{1}{C_n^2}$$

- După C_n^2 încercări probabilitatea de eșec este $\le e^{-1}$
- După $C_n^2 \ln(n)$ încercări probabilitatea de eșec este $\le e^{-\ln(n)} = \frac{1}{n}$
- După $r \cdot C_n^2 \ln(n)$ încercări probabilitatea de eșec este $\le \frac{1}{n^r}$

Algoritmi Monte Carlo

Algoritmul lui SCHÖNING pentru 3-SAT

- n variabile +negații
- $E = C_1 \wedge C_2 \wedge \dots \wedge C_m$ – presupunem satisfiabilă
- **Clauze C_i disjunctive – cu cel mult 3 literali**

  - **$O(2^n m)$ – încercând toate cele 2^n asocieri posibile**

Algoritmul lui SCHÖNING pentru 3-SAT

Algoritmul lui SCHÖNING pentru 3-SAT (vers1)

O etapă (se repetă de un număr de ori):

1. Asociază aleatoriu valori variabilelor $x=(x_1,\ldots, x_n)$
2. Repetă de n ori
   - Dacă toate clauzele sunt satisfăcute STOP
   - Alege aleatoriu o clauză C nesatisfăcută
   - Alege aleatoriu o variabilă din C și modifică-i valoarea (true$\leftrightarrow$false)

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- Fie $v = (v_1,\ldots,v_n)$ o asociere de valori pentru variabilele **pentru care expresia este adevărată** (soluție corectă)

- Studiem pe parcursul algoritmului

  dist(x,v)=numărul de poziții pe care x și v diferă

  (distanța Hamming)

(pentru a determina probabilitatea ca x să devină egal cu v)

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**

  "Scenariu"

  - Inițial: **dist(x, v) ≤ n/2**
  - La o repetare distanța d(x, v) scade cu 1

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**
  - Fie **A evenimentul**: după pasul 1 **dist(x,v) ≤ n/2**

$$\Pr[A] \geq ?$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**
  - Fie **A evenimentul**: după pasul 1 **dist(x,v) ≤ n/2**

$$\Pr[A] \geq \frac{1}{2}$$

    - **Simetrie: numărul de șiruri x cu dist(x,v) = k**

$$= \; C_n^k = C_n^{n-k}$$

= numărul de șiruri x cu dist(x,v) = n-k

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**
  - La **o trecere** prin ciclul repeat **probabilitatea ca** dist(x,v) să scadă cu 1 este $\geq$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**
  - La o trecere prin ciclul repeat **probabilitatea ca** dist(x,v) să scadă cu 1 este $\geq 1/3$
  - **sunt cel mult n/2 poziții pe care x și v diferă**
  - $\Rightarrow$ **probabilitatea de succes p este**

$$p \geq \Pr[A] \cdot \Pr[succes \mid A] \geq$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **O primă analiză:**
  - La o trecere prin ciclul repeat **probabilitatea ca** dist(x,v) să scadă cu 1 este $\geq 1/3$
  - **sunt cel mult n/2 poziții pe care x și v diferă**
  - $\Rightarrow$ **probabilitatea de succes p este**

$$p \geq \Pr[A] \cdot \Pr[succes \mid A] \geq$$

$$\geq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{\frac{n}{2}}$$

Algoritmul lui SCHÖNING pentru 3-SAT

- **Probabilitatea de succes?**
- **O primă analiză:**

$$p \geq \frac{1}{2}\left(\frac{1}{3}\right)^{\frac{n}{2}}$$

După k repetări, probabilitatea de eșec va fi

$$(1-p)^k \leq e^{-pk}$$

Algoritmul lui SCHÖNING pentru 3-SAT

- **Probabilitatea de succes?**

- **O primă analiză:**

$$p \geq \frac{1}{2}\left(\frac{1}{3}\right)^{\frac{n}{2}}$$

După k repetări, probabilitatea de eșec va fi

$$(1-p)^k \leq e^{-pk}$$

Pentru $k = \frac{\ln n}{p} \cdot r$ probabilitatea de eșec va fi $\leq \frac{1}{n^r}$

$$p \geq \frac{1}{2}\left(\frac{1}{3}\right)^{\frac{n}{2}} \Rightarrow k \leq 2r\left(\sqrt{3}\right)^n \ln(n)$$

$$O(\left(\sqrt{3}\right)^n \ln(n))$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Analiza 2:**
  - Fie A_k evenimentul: după pasul 1 (de inițializare) x și v diferă **pe exact k poziții**:

    dist(x,v) = k

$$\Pr[A_k] = ?$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Analiza 2:**
  - Fie A_k evenimentul: după pasul 1 (de inițializare) x și v diferă **pe exact k poziții**:

    dist(x,v) = k

  - $$\Pr[A_k] = \binom{n}{k} \cdot \left(\frac{1}{2}\right)^k \cdot \left(\frac{1}{2}\right)^{n-k} = \binom{n}{k} \cdot \left(\frac{1}{2}\right)^n$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Analiza 2:**

Probabilitatea de succes

$$p \geq \sum_{k=0}^{n} \Pr[A_k] \cdot \Pr[succes \mid A_k]$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Analiza 2:**

Probabilitatea de succes

$$p \geq \sum_{k=0}^{n} \Pr[A_k] \cdot \Pr[succes \mid A_k] =$$

$$= \sum_{k=0}^{n} \binom{n}{k} \cdot \left(\frac{1}{2}\right)^n \left(\frac{1}{3}\right)^k$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Analiza 2:**

Probabilitatea de succes

$$p \geq \sum_{k=0}^{n} \Pr[A_k] \cdot \Pr[succes \mid A_k] =$$

$$= \sum_{k=0}^{n} \binom{n}{k} \cdot \left(\frac{1}{2}\right)^n \left(\frac{1}{3}\right)^k = \left(\frac{1}{2}\right)^n \sum_{k=0}^{n} \binom{n}{k} \cdot \left(\frac{1}{3}\right)^k =$$

$$= \left(\frac{1}{2}\right)^n \cdot \left(1 + \frac{1}{3}\right)^n = \left(\frac{2}{3}\right)^n$$

Algoritmul lui SCHÖNING pentru 3-SAT

- **Probabilitatea de succes?**

- **Analiza 2:**

$$p \geq \left(\frac{2}{3}\right)^n$$

După k repetări, probabilitatea de eșec va fi

$$(1-p)^k \leq e^{-pk}$$

Pentru $k = \frac{\ln n}{p} \cdot r$ probabilitatea de eșec va fi $\leq \frac{1}{n^r}$

$$p \geq \left(\frac{2}{3}\right)^n \Rightarrow k \leq r\left(\frac{3}{2}\right)^n \ln(n)$$

$$O(\left(\frac{3}{2}\right)^n \ln(n)) \Rightarrow O((1,5)^n \ln(n))$$

Algoritmul lui SCHÖNING pentru 3-SAT

Probabilitatea de succes?

- **Varianta îmbunătățită – repet de 3n ori**

Analiza

http://www.comp.nus.edu.sg/~rahul/allfiles/cs6234-16-random-3sat.pdf

https://www.cs.cmu.edu/~15210/recitations/Randomized3Sat.pdf

U. Schöning: **A probabilistic algorithm for k-SAT and constraint satisfaction problems**, Proc. 40th Annual Symp. Foundations of Computer Science, IEEE Computer Society (1999), 410 -414

Algoritmi Las Vegas

Algoritmi Las Vegas

- **Nu furnizează totdeauna un rezultat**, dar dacă furnizează un rezultat atunci acesta este **corect**

- **Probabilitatea ca algoritmul să se termine creşte pe măsură ce timpul disponibil creşte**

```
repeat
   if LV()
        stop
until false
```

Algoritmi Las Vegas

Se dau n texte (n foarte mare) cu următoarele proprietăţi:

- **există un unic text t_0 care apare de cel puţin 10% ori;**
- **celelalte texte sunt distincte.**

Se cere determinarea textului t_0.

Algoritmi Las Vegas – Text

Algoritm probabilist

Idee

- Generăm aleatoriu doi indici i și j și testăm dacă

  $i \neq j$ și $t_i = t_j$

până când testul se încheie cu succes

Algoritmi Las Vegas – Text

```
repeat
   if LVText()
       stop
until false


LVText()
     i ← random(1..n); j ← random(1..n);
     if i≠j and t_i=t_j
         write t_i; return true
     else
         return false
```

Algoritmi Las Vegas - Text

Probabilitatea p de succes la un pas

(**`LVText`**() returnează true):

$=$ probabilitatea p ca $t_i = t_j = t_0$, $j \neq i$

p = ?

Algoritmi Las Vegas – Text

Probabilitatea p de succes (**`LVText`**() returnează true):

= probabilitatea p ca $t_i = t_j = t_0$, $j \neq i$

- probabilitatea ca $t_i = t_0$
- Dacă $t_i = t_0$, probabilitatea ca $t_j = t_0$, $j \neq i$

Algoritmi Las Vegas – Text

Probabilitatea p de succes (**`LVText`**() returnează true):

= probabilitatea p ca $t_i = t_j = t_0$, $j \neq i$

- probabilitatea ca $t_i = t_0 \Rightarrow \frac{\frac{1}{10}n}{n} = 1/10$
- probabilitatea ca $t_j = t_0$, $j \neq i \Rightarrow \frac{\frac{1}{10}n-1}{n} = 1/10 - 1/n$

(dacă $t_i = t_0$)

Algoritmi Las Vegas – Text

Probabilitatea p de succes (**`LVText`**() returnează true):

= probabilitatea p ca $t_i = t_j = t_0$, $j \neq i$

- probabilitatea ca $t_i = t_0 \Rightarrow \frac{\frac{1}{10}n}{n} = 1/10$
- probabilitatea ca $t_j = t_0$, $j \neq i \Rightarrow \frac{\frac{1}{10}n-1}{n} = 1/10 - 1/n$

(dacă $t_i = t_0$)

$$p = \frac{1}{10}\left(\frac{1}{10} - \frac{1}{n}\right) \geq \frac{9}{1000} \text{ pentru } n \geq 100$$

Algoritmi Las Vegas – Text

Probabilitatea p de succes (**`LVText`**() returnează true):

= probabilitatea p ca $t_i = t_j = t_0$, $j \neq i$

- probabilitatea ca $t_i = t_0 \Rightarrow \frac{\frac{1}{10}n}{n} = 1/10$
- probabilitatea ca $t_j = t_0$, $j \neq i \Rightarrow \frac{\frac{1}{10}n-1}{n} = 1/10 - 1/n$

(dacă $t_i = t_0$)

$$p = \frac{1}{10}\left(\frac{1}{10} - \frac{1}{n}\right) \geq \frac{9}{1000} \text{ pentru } n \geq 100$$

Teoretic sunt suficiente $t = 1/p \approx$ **112 încercări**

(apeluri ale **`LVText`**())

Algoritmi Las Vegas

- **Analiza timpului estimat pentru răspuns cu succes**

```
repeat
   if LV()
        stop
until false
```

- s = timpul mediu a unei rulări cu succes a **`LV()`**
- f = timpul mediu a unei rulări cu eșec a **`LV()`**
- p = probabilitatea de succes
- t = **timpului estimat pentru răspuns cu succes** (repetând funcția **`LV()`**) = ?

Algoritmi Las Vegas

- **Analiza timpului estimat pentru răspuns cu succes**

```
repeat
   if LV()
        stop
until false
```

- s = timpul mediu a unei rulări cu succes a `LV()`
- f = timpul mediu a unei rulări cu eșec a `LV()`
- p = probabilitatea de succes
- t = **timpului estimat pentru răspuns cu succes** (repetând funcția `LV()`)

$$t = p \cdot s + (1-p) \cdot (f+t) \Rightarrow t = s + f \cdot (1-p)/p$$

Algoritmi Las Vegas

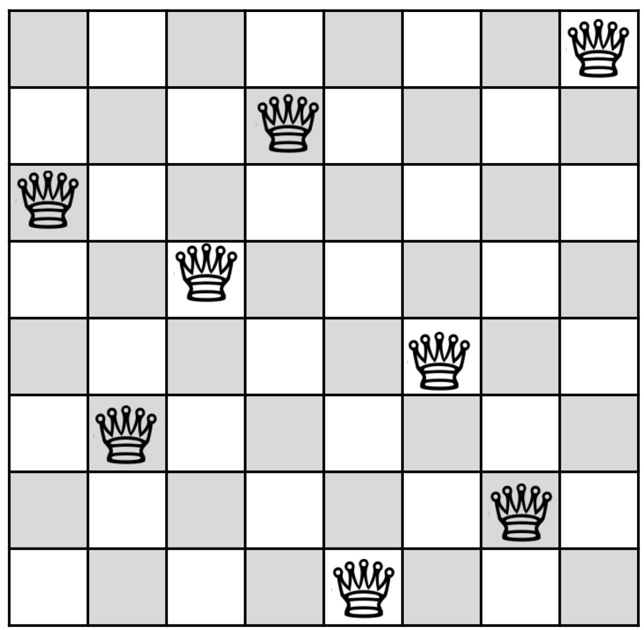
- **Analiza timpului estimat pentru răspuns cu succes**
  - Pentru $s = f$ (cum este cazul `LVText()`) obținem

$$t = s \cdot 1/p$$

Algoritmi Las Vegas

Problema celor n dame

Se consideră un caroiaj n×n. Prin analogie cu o tablă de şah (n=8), se doreşte plasarea a n dame pe pătrăţelele caroiajului, astfel încât să nu existe două dame una în bătaia celeilalte



Algoritmi Las Vegas – Problema damelor

- **Reprezentarea soluţiei**

  $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, unde

  $\mathbf{x}_k$ = coloana pe care este plasată dama de pe linia $\mathbf{k}$

  $\mathbf{x}_k \in \{1, 2, \ldots, n\}$

Algoritmi Las Vegas – Problema damelor

- **Backtracking**

  `n = 8` – explorate 114 vârfuri (din 2057)

  din arborele asociat spațiului de căutare

  până când este găsită prima soluție

Algoritmi Las Vegas – Problema damelor

Algoritm probabilist

Idee

- pornim de la prima linie

repetă

- plasăm o damă aleatoriu pe linia curentă
- dacă dama nu atacă nicio damă deja plasată se trece la linia următoare

  altfel

până când se ajunge la linia n+1

Algoritmi Las Vegas – Problema damelor

Algoritm probabilist

Idee

- pornim de la prima linie

repetă

- plasăm o damă aleatoriu pe linia curentă
- dacă dama nu atacă nicio damă deja plasată se trece la linia următoare

  altfel **eșec** (return false) **reluăm întreg algoritmul, nu facem doar un pas înapoi ca la Backtracking** (linia curentă devine prima linie)

până când se ajunge la linia n+1

Algoritmi Las Vegas - Problema damelor

Cum gestionăm diagonalele și coloanele pe care s-au plasat deja dame?

Algoritmi Las Vegas – Problema damelor

Vectorii:

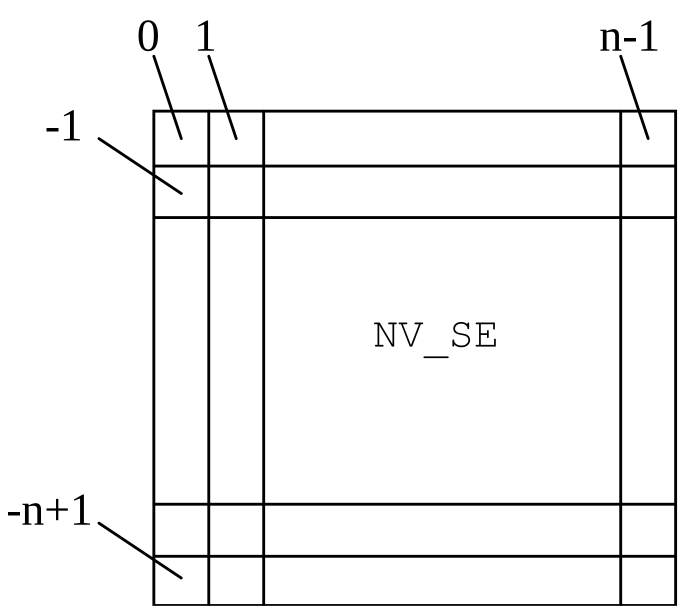
- Diagonale paralele cu diagonala principală (j – i = **constant**)

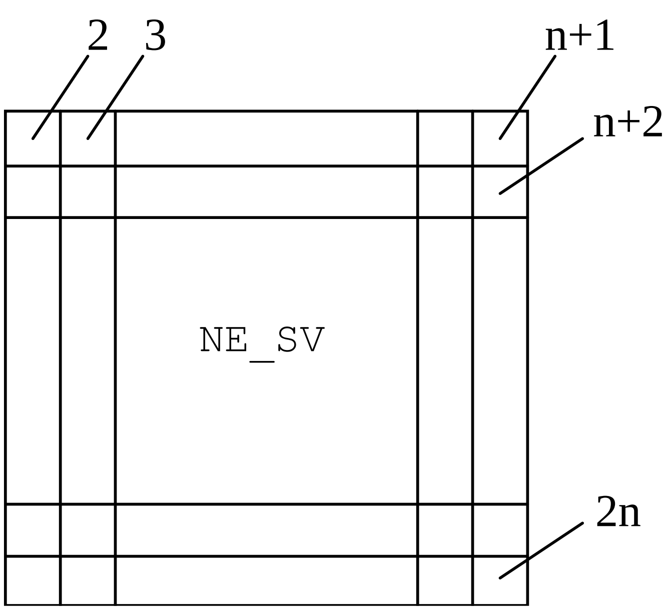
  NV_SE[−n+1..n−1]

- Diagonale paralele cu diagonala secundară (j + i = **constant**)

  NE_SV[2..2n]

- Coloane **C[1..n]**





Algoritmi Las Vegas – Problema damelor

```
repeat
    if LVDame() then
        stop
until false
```

```
LVDame()
    • iniţializăm C, NV_SE, NE_SV cu true
    • k ← 1
    repeat
      •

      •

    until k=n+1
    write(x)
    return true
```

LVDame()

- **iniţializăm** C, NV_SE, NE_SV cu true
- k ← 1

repeat

- formăm un vector a cu acele poziţii i∈{1,...,n} cu **C[i] = NV_SE[i-k] = NE_SV[i+k] = true** şi notăm na lungimea vectorului obţinut
-

until **k=n+1**

write(x)

return true

```
LVDame()
    • iniţializăm C, NV_SE, NE_SV cu true
    • k ← 1
    repeat
      • formăm un vector a cu acele poziţii i∈{1,...,n}
        cu C[i] = NV_SE[i-k] = NE_SV[i+k] = true
         şi notăm na lungimea vectorului obţinut
      • if na>0 then
            aleg aleator i∈{1,...,na};  poz ← a_i
            x_k ← poz ;{plasam aleator dama pe o pozitie corecta}




    until k=n+1
    write(x)
    return true
```

LVDame()

```
• iniţializăm C, NV_SE, NE_SV cu true
• k ← 1
repeat
  • formăm un vector a cu acele poziţii i∈{1,...,n}
    cu C[i] = NV_SE[i-k] = NE_SV[i+k] = true
     şi notăm na lungimea vectorului obţinut
  • if na>0 then
        aleg aleator i∈{1,...,na};  poz ← a_i
        x_k ← poz ;
        NV_SE[poz-k] ← false; NE_SV[poz+k] ← false;
        C[poz] ← false;
        k ← k+1
    else

 until k=n+1
 write(x)
 return true
```

```
LVDame()
      • iniţializăm C, NV_SE, NE_SV cu true
      • k ← 1
      repeat
        • formăm un vector a cu acele poziţii i∈{1,...,n}
          cu C[i] = NV_SE[i-k] = NE_SV[i+k] = true
          şi notăm na lungimea vectorului obţinut
        • if na>0 then
              aleg aleator i∈{1,...,na};  poz ← a_i
              x_k ← poz ;
              NV_SE[poz-k] ← false; NE_SV[poz+k] ← false;
              C[poz] ← false;
              k ← k+1
          else
              return false
      until k=n+1
      write(x)
      return true
```

Algoritmi Las Vegas – Problema damelor

- **Analiza probabilitate succes**
  - p = probabilitatea de succes
  - s = numărul mediu de vârfuri explorate la un succes
  - f = numărul mediu de vârfuri explorate la un eșec
  - t = numărul de vârfuri explorate până la încheierea cu succes (repetând funcția LVDame())

$$\mathbf{t = s + f \cdot (1-p)/p}$$

Algoritmi Las Vegas – Problema damelor

- **Experimente $n = 8$**
  - $p \approx 0.1293$
  - $f \approx 6.971$
  - $s = 9$
  - $t = s + f \cdot (1-p)/p \approx 56$

Algoritmi Las Vegas – Problema damelor

- **Experimente $n = 8$**
  - $p \approx 0.1293$
  - $f \approx 6.971$
  - $s = 9$
  - $t = s + f \cdot (1-p)/p \approx 56$

- **Backtracking –114**

- **Soluții mixte – k dame plasate aleatoriu, apoi backtracking**

Algoritmi numerici

Algoritmi numerici

- Urmăresc determinarea aproximativă a unei valori
- **Cu cât timpul alocat executării algoritmului este mai mare, precizia rezultatului se îmbunătăţeşte**

Algoritmi numerici

- **Aproximarea** lui π
- **Aproximarea** $\int_a^b f(x)\,dx$

$$f:[a,b]\to[c,d]$$

Aproximarea lui π

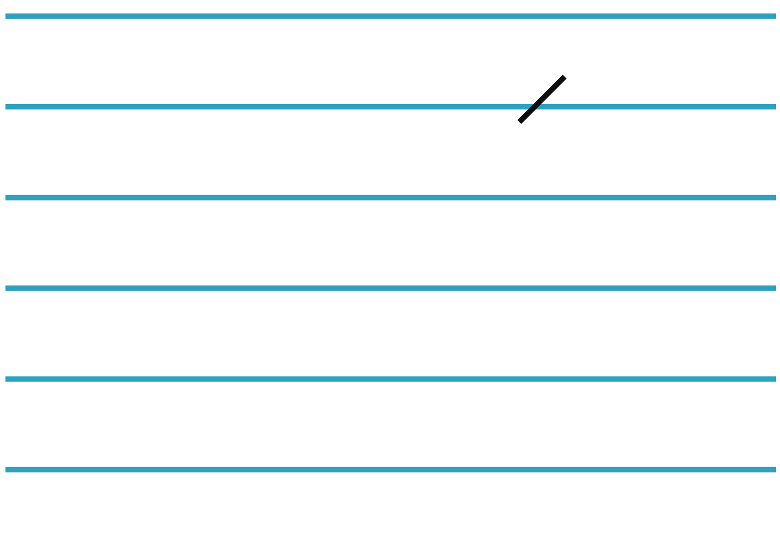
1. Acul lui Buffon

Se consideră o mulţime de linii paralele astfel încât oricare două linii vecine sunt la distanţă de o unitate.

Aproximarea lui π

1. Acul lui Buffon

Se consideră o mulţime de linii paralele astfel încât oricare două linii vecine sunt la distanţă de o unitate. Un ac de lungime o jumătate de unitate este aruncat aleator şi se numără de câte ori a intersectat vreo linie



Aproximarea lui π

1. Acul lui Buffon

- Probabilitatea ca acul să intersecteze o linie este $1/\pi$

Aproximarea lui π

1. Acul lui Buffon

- **Probabilitatea ca acul să intersecteze o linie este $1/\pi$**
- După un număr "suficient de mare" de încercări, raportul între:

    - **`numărul total de încercări`**

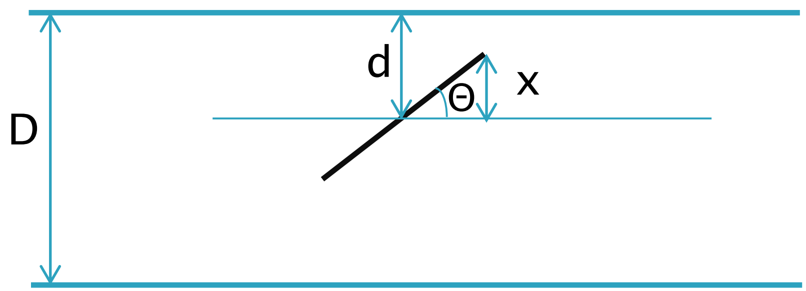
    - **`numărul cazurilor în care acul a intersectat vreo linie`**

va fi "suficient de aproape" de π.

Aproximarea lui π

1. Acul lui Buffon

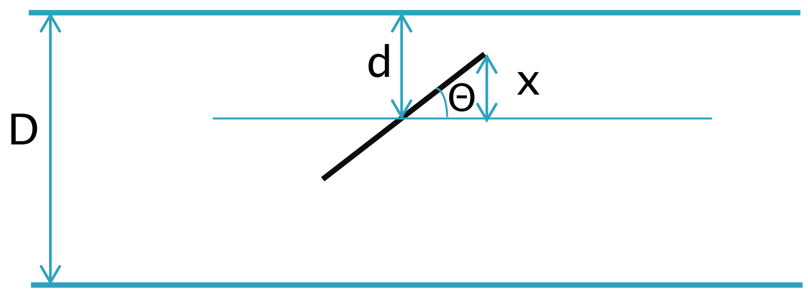
- **Justificare:**
  - L – lungimea acului (exp L=1/2)
  - D – distanța dintre drepte (L<D, exp D=1)
  - Acul – unic identificat de perechea (Θ,d), unde



Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - L – lungimea acului (exp L=1/2)
  - D – distanța dintre drepte (L<D, exp D=1)
  - Acul – unic identificat de perechea (Θ,d), unde
    - d = distanța de la centrul acului la cea mai apropiată dreaptă (linie) din mulțime
    - Θ = unghiul format de ac cu direcția dreptelor paralele

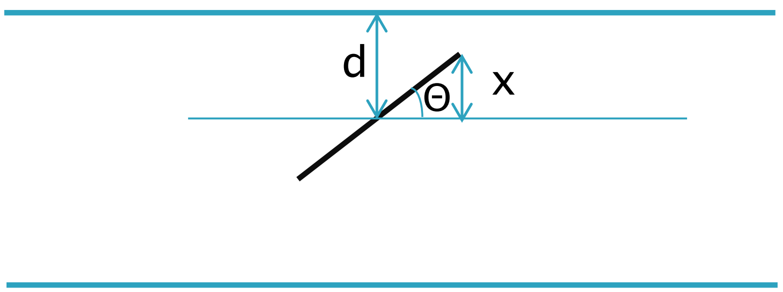


Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - Acul – unic identificat de perechea (Θ,d), unde
    - d $\in$[0, D/2]
    - Θ $\in$[0, π]
  - **“Aruncare ac” ⇔ generare pereche (Θ,d)** / (sin(Θ),d)
  - Acul intersectează dreapta cea mai apropiată ⇔

    d ≤ x=L/2 sin(Θ)



Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - Poziții **posibile ac**:
    - $T = \{(\Theta, d) \mid d \in [0, D/2], \Theta \in [0, \pi]\}$

Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - Poziții **posibile ac**:
    - $T = \{(\Theta,d) \mid d \in [0, D/2], \Theta \in [0, \pi] \}$
  - Poziții **ac – care intersectează dreaptă**:
    - $F = \{(\Theta,d) \mid \Theta \in [0, \pi], 0 \le d \le L/2 \sin(\Theta) \}$

Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - Poziții **posibile ac**:
    - $T = \{(\Theta,d) \mid d \in [0, D/2], \Theta \in [0, \pi] \}$
  - Poziții **ac – care intersectează dreaptă**:
    - $F = \{(\Theta,d) \mid \Theta \in [0, \pi], 0 \le d \le L/2 \sin(\Theta) \}$
  - Probabilitatea ca acul să intersecteze dreapta:

$$\frac{arie(F)}{arie(T)} =$$

Aproximarea lui π

1. Acul lui Buffon

- **Justificare:**
  - Poziții **posibile ac**:
    - $T = \{(\Theta,d) \mid d \in [0, D/2], \Theta \in [0, \pi]\}$
  - Poziții **ac – care intersectează dreaptă**:
    - $F = \{(\Theta,d) \mid \Theta \in [0, \pi], 0 \leq d \leq L/2 \sin(\Theta)\}$
  - Probabilitatea ca acul să intersecteze dreapta:

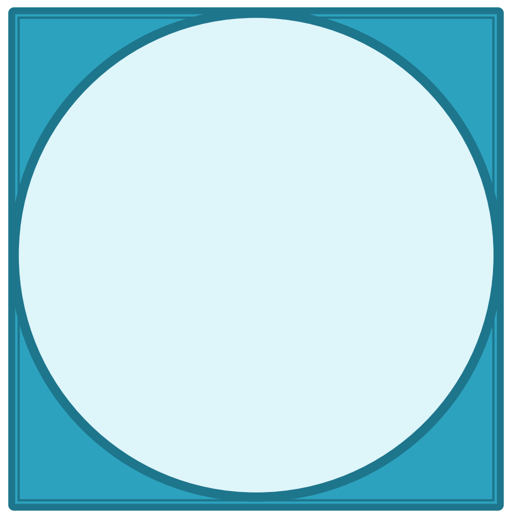
$$\frac{arie(F)}{arie(T)} = \frac{\left|\int_0^{\pi} \frac{L}{2} \sin(\Theta)\, d\Theta\right|}{\pi D/2} = \frac{2L/2}{\pi D/2} = \frac{2L}{\pi D}$$

Pentru L=1/2 și D=1 probabilitatea este $\frac{1}{\pi}$

Aproximarea lui π

2. Se aruncă repetat cu o săgeată într-un panou pătrat, cu ținta un cerc înscris în pătrat.

Se presupune că săgeata nimereşte totdeauna panoul.



Aproximarea lui π

2. Se aruncă repetat cu o săgeată într-un panou pătrat, cu ținta un cerc înscris în pătrat.

Se presupune că săgeata nimereşte totdeauna panoul.

Atunci raportul dintre:

- numărul cazurilor în care săgeata nimereşte în cercul înscris în pătrat

- numărul total de încercări

tinde la

$$\frac{arie\ cerc}{arie\ patrat} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

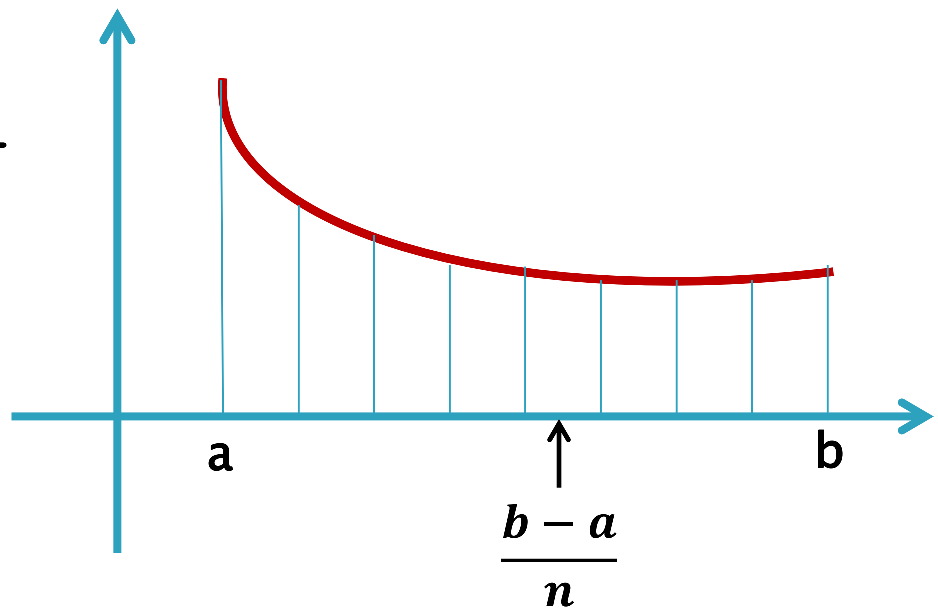
Aproximarea integralei

$$\int_a^b f(x)\,dx, \quad f:[a,b] \rightarrow [c,d]$$

Aproximarea integralei

$$\int_a^b f(x)\,dx, \qquad f:[a,b]\to[c,d]$$

n încercări



Aproximarea integralei

$$\int_a^b f(x)\,dx, \quad f:[a,b]\to[c,d]$$

```
s ← 0
for i=1,n
    x ← random([a,b]);
    s ← s+f(x)

s ← s·(b-a)/n
write(s)
```