CS 598: Pseudorandomness; Lecture 6

Today: conditional expectations, pairwise independence.

Max cut

\[ G = (V, E) \quad V = S \cup T \quad \text{cut}(S, T) = \# \text{edges between } S, T \]

The \( \frac{1}{2} \) approx maxcut in randomized polytime:

- [ ] edge, draw \( 1 \epsilon + 1 \) random cuts
- choose best cut

Correctness:

\[ E[cut] = \epsilon / 2 \]

Markov:

\[ P(\text{cut} \geq \epsilon) \leq \frac{1}{e+1} \]

Time:

\[ \text{work } O(n^2) \]

\[ \text{expected time } \text{poly}(|E, V|) \]

\[ \text{space } 2(n) \]

\[ \text{need to store random bit} \]

Q: Cond. expectations?

Rephrase:

- Pick \( R_1, \ldots, R_n \) random

\[ S = \{ e : R_i = 0 \} \]

\[ T = \{ e : R_i = 1 \} \]

Q: Any cut is good?

\[ P(r_1, \ldots, r_n) = E[\text{cut} | R_i = r_i \quad \forall i] \]

\[ P(r_1, \ldots, r_n) = \prod_{i=1}^{n} P(r_i) \]

\[ P(r_1, \ldots, r_n) = \frac{1}{2^n} \]

\[ P(r_1, \ldots, r_n) = \text{size or single cut} \]
Lemma: For $i, j$ random vars $E_X[f_i] = E_Y[f_j]$

$$P(\pi_i, \ldots, \pi_i) = E_{\pi_i, \ldots, \pi_i} P(\pi_i, \ldots, \pi_i, R_i)$$

$$= \frac{1}{2} P(\pi_i, \pi_j, R_i) + \frac{1}{2} P(\pi_i, \pi_j, R_i, R_i)$$

$$\Rightarrow \forall \pi_i, \pi_j \quad P(\pi_i, \pi_j, R_i, R_i) = P(\pi_i, \pi_j, R_i)$$

Then define $Z$

Lemma: For $i, j$ random vars $S_i = S_j = 0$

$$T_i = \left[ \begin{array}{c} i = 0 \\ i = 1 \end{array} \right]$$

then $P(\pi_i, \ldots, \pi_i) = \text{cut}(S_i, T_i) + \frac{1}{2} \text{cut}(V - (S_i, T_i), V)$

$$P_i = \left[ \begin{array}{c} \text{cut}(S_i, T_i) + \frac{1}{2} \text{cut}(V - (S_i, T_i), V) \\ \text{cut}(S_i, T_i) + \frac{1}{2} \text{cut}(V - (S_i, T_i), V) \end{array} \right]$$

Remark: Easily computed

- $P_i$ is independent of $\pi_i, \ldots, \pi_i$

Lemma: $P(\pi_i, \pi_i, \pi_j, \pi_i) = \text{cut}(S_i, S_i) - \text{cut}(T_i, S_i)$

Case: $S_i, T_i = 0$

- $\text{cut}(S_i, S_i) > \text{cut}(T_i, S_i)$, put $i = 1$

else

Correctness: Clear

TODO: $O(n)$ worse

Parallelism is many sequential choices

Small space: Need to store these choices

Q: Can we parallelize / small space?

Question: Remark: This wasn't really about pseudorandomness, it was whether...
def \( h : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \) is a *pairwise independent hash function*

\[ \forall i, j \in \mathbb{N} : h(i, \cdot) \sim h(j, \cdot) \]

is *computable* in \( \text{poly}(\log N, \log h) \) time.

Rmk: hash functions are practical & resemble random functions, but

Q: how to construct

recall finite fields:

- If field \( \mathbb{F} = \text{addition, multiplication} \)
- Example: \( \mathbb{F} = \mathbb{F}_2 = \{0, 1\} \)
- \( \mathbb{F}_2 \) field, \( \mathbb{F}_{2^n} \) field

\[ \mathbb{F}_{2^n} = \mathbb{F}_2^n \]

is **pairwise in dependent**

\[ h : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2 \]

and explicit: \( \text{this is clear} \mathbb{Z} \)

Proof:

- \( h : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2 \)
- \( h(x, y) = \alpha \wedge \beta \) & \( \# \text{lines through } (x, \alpha), (y, \beta) \) is \( \frac{1}{\mathbb{F}_2} \)

**Linear probing**

**Hash tables**

\( h : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2 \)

\( h(x, y) = \alpha \wedge \beta \) & \( \# \text{lines through } (x, \alpha), (y, \beta) \) is \( \frac{1}{\mathbb{F}_2} \)

\( \mathbb{F}_2 \) vs \( \mathbb{F}_{2^n} \) random bits for \( n \)

Car:

- \( h : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2 \)
- \( h(x, y) = \alpha \wedge \beta \) & \( \# \text{lines through } (x, \alpha), (y, \beta) \) is \( \frac{1}{\mathbb{F}_2} \)

**Car:** \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log \log n) \) \( \mathcal{O}(n \log n) \)

Proof: \( \mathcal{O}(n \log n) \) pam of \( \mathcal{O}(n \log n) \) pam

\( \mathcal{O}(n \log n) \) pam of \( \mathcal{O}(n \log n) \) pam

**Car:** \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \)

Proof: \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \)

**Car:** \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \)

Proof: \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \) \( \mathcal{O}(n \log n) \)
next step, k w/ independence

amazing sample