**Today:** 1:15 - 2:15 pm

Final exam: check date/time on classpop & piazza.

**Skip Lists**

**Thm:** With a high probability, every search in a skip list with n elements takes \( O(\log n) \) time.

**Proof:** When we are searching for an element \( x \) in the skip list
- we start at the top level from the root node.
- we follow down links or the right links.
we stop at the largest element \( \leq x \) in \( L_1 \).

Total Cost: Down links + Right links. Instead, we will start at \( v_i \) and retrace the path in the reverse direction, i.e., we go up or left.
Total cost: Up links + left links.

\[ \# \text{HEADS} + \# \text{TAILS}. \]

From the prev class, we know that the height of the sleep list is \( \leq 2 \log n \), with a prob \( 1 - \frac{1}{n} \). That is, the prob with which the height is \( > 2 \log n \leq \frac{n}{n} \).

Thus \( \# \text{HEADS} > 2 \log n \) with prob \( \leq \frac{1}{n} \).

We will prove that it is highly unlikely that after \( 20 \log n \) coin tosses we have not reached the -oo node in \( L_g \).

E: event that at the end of \( 20 \log n \) coin tosses, our search has not ended.
$U$: event that the #Heads \geq 21\lg n.$

\[ P[E] = P[E \cap U] + P[E \cap \overline{U}] \]

\[ \leq P[U] + P[E] \cdot P[\overline{U} | E] \]

\[ \leq \left[ \frac{1}{n} + P[\overline{U} | E] \right] \]

\[ P[\overline{U} | E] = \sum_{i=0}^{21\lg n} \left( \begin{array}{c} 20\lg n \\ i \end{array} \right) \cdot \left( \frac{1}{2} \right)^i \left( \frac{1}{2} \right)^{20\lg n - i} \]

\[ = \left( \frac{1}{2} \right)^{20\lg n} \sum_{i=0}^{21\lg n} \left( \begin{array}{c} 20\lg n \\ i \end{array} \right) \]

\[ \leq 2^{-20\lg n} \cdot \left( \frac{20\lg n \cdot e}{21\lg n} \right)^{21\lg n} \]

The above inequality follows from

\[ \sum_{i=0}^{r} \left( \begin{array}{c} n \\ i \end{array} \right) \leq \left( \frac{ne}{r} \right)^n. \]

\[-20\lg n \quad 21\lg n\]
\[
\begin{align*}
\frac{2^{\text{loge}^2}}{20 \ln n} & \geq \frac{\ln 10e - 2 \ln n}{2} \\
\ln n & \approx \frac{\ln (10e - 10)}{2} \quad \leftarrow -1 \\
\frac{1}{2 \ln n} & = \frac{1}{n^2} \\
\Pr(\mathcal{E}) & \leq \frac{1}{n} + \frac{1}{n^2} \\
& \leq \frac{1}{n} + \frac{1}{n} \\
& = \frac{2}{n}
\end{align*}
\]

Mincuts.
Input: Undirected graph $G = (V, E)$

Obj: Partition $V$ into $V_1$ and $V_2$ s.t. 
the number of edges crossing $V_1$ and $V_2$ is minimized.

Karger's Algorithm:
- Pick an edge at random.
- Contract it.
- Repeat the above steps until we have two nodes left.

Analysis:
$C$: my favorite min-cut.
| \(|C| = k \quad (\text{\# edges that cross the cut})

E_i: \text{event that cut } C \text{ is safe in iteration } i.

We want to find

\[ \Pr [ E_1 \cap E_2 \cap \ldots \cap E_{n-2}] \]

\[ = \Pr [ E_1] \cdot \Pr [ E_1 | E_2] \cdot \Pr [ E_2 | E_1 \cap E_2] \cdot \ldots \cdot \]

\[ \Pr [ E_1] = \frac{k}{m} \leq \frac{k}{\frac{n \cdot k}{2}} = \frac{2}{n} \cdot \left[ m \geq \frac{n \cdot k}{2} \right] \left[ \delta \geq k \right] \]

\[ \frac{m}{\sum y(v)} \]

\[ \Pr [ E_1] \geq 1 - \frac{2}{n} = \frac{n-2}{n}. \] (a)

\[ \Pr [ E_2 | E_1] = \frac{k}{m_1} \leq \frac{2}{n-1} \left[ \delta_1 \geq k \right] \left[ m_1 \geq \frac{(n-1) \cdot k}{2} \right] \]

\[ \therefore \Pr [ E_1 | E_2] \geq 1 - \frac{2}{n} = \frac{n-2}{n}. \] (b)
\[ P \left( \bigcap_{i=1}^{n-1} E_i \right) \geq \left( \frac{n-2}{n} \right) \cdot \left( \frac{n-3}{n-1} \right) \cdot \left( \frac{n-4}{n-2} \right) \cdots \left( \frac{3}{5} \right) \left( \frac{2}{4} \right) \left( \frac{1}{3} \right) \] 

\[ = \frac{2}{n(n-1)} \]

Run the algo. \( p \) times & output the smallest of all the cuts obtained.

\[ \left( 1 - \frac{2}{n(n-1)} \right)^p \leq e^{-\frac{2}{n(n-1)} \cdot p} \]

\( (1 + x \leq e^x) \)
Set \( p = \frac{n(n-1)}{2} \) in \( n \).

\[ \Pr[\text{bad event}] \leq e^{-\ln n} = \frac{1}{n}. \]

Running time: \( \approx n^4 \)

\[ \text{(Karlin-Slevin).} \]

3-2-SAT

Input: \( \phi_{n,m} \)

\[ \begin{align*}
& (x_1 \lor \overline{x}_2) \\
\text{clause} \\
& (x_2 \lor x_3) \land (x_2 \lor \overline{x}_4) \land (x_3 \lor x_5) \land \ldots
\end{align*} \]

Obj: Is \( \phi \) satisfiable?

1. Start with an arbitrary assign.
2. If $\phi$ is satisfiable:
   Done.

3. Else
   pick a clause that is not satisfied & uniformly at random pick one of its literals and flip its boolean value.

4. Keep doing the above until we get a fixed $\sim n^2$. 