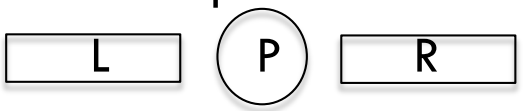
RANDOMIZED ALGORITHMS

Monte Carlo

Las Vegas

Introduction - Quicksort

- Algorithm
  - Input: S = { x1,x2,……., xn}
  - Quicksort(S)
    1. Randomly pick a pivot element p ← x1 …… xn
    2. Partition S about p 

       where L ≤ p ≤ R
    3. Quicksort(L)
    4. Quicksort(R)

Let T(n) be the running time for this algorithm

Xij -: indicates whether I was compared to j

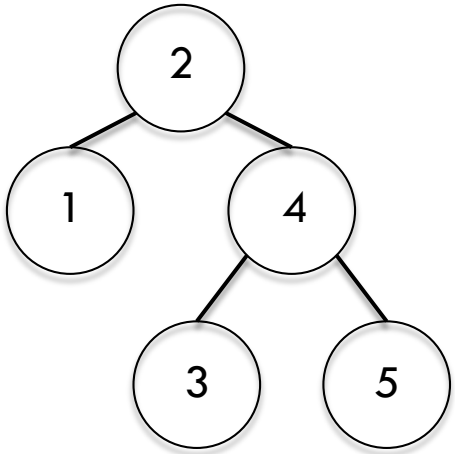
	X_{12}	X_{13}	…….....	…….....	Accumulator T(n)
1st run	1	0	.	.	84
2nd run	1	1	.	.	79
.	.	.	.	.	.
.	.	.	.	.	.

E(T(n)) = Average of all T(n)

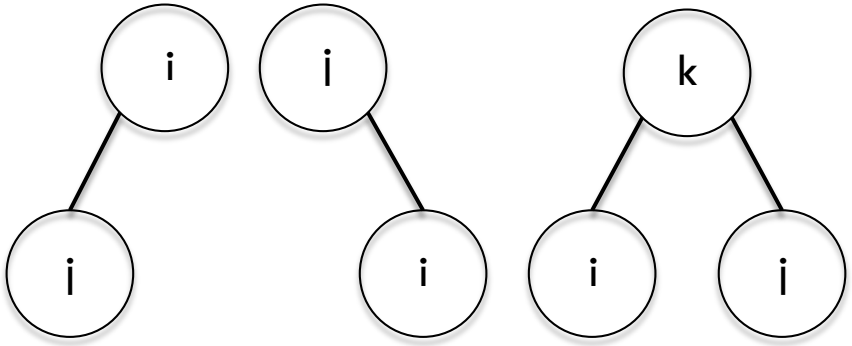
$$T(n) = \sum_{i=1}^{n} \sum_{j>i}^{n} Xij$$

$$E[\,T(n)\,] = E\left[\; \sum_{i=1}^{n} \sum_{j>i}^{n} Xij \;\right]$$

$$= \sum_{i=1}^{n} \sum_{j>i}^{n} E[\,Xij\,]$$



- i and j are compared iff one end up as a descendent of the other
- P_{ij} = probability that i is compared with j = $E[X_{ij}]$



3 possibles ways I and j can occur in the tree

$$P_{ij} = \frac{2}{j - i + 1}$$

$$j - I + 1 = k$$

j	k
i+1	2
i+2	3
.	.
n	n-i+1

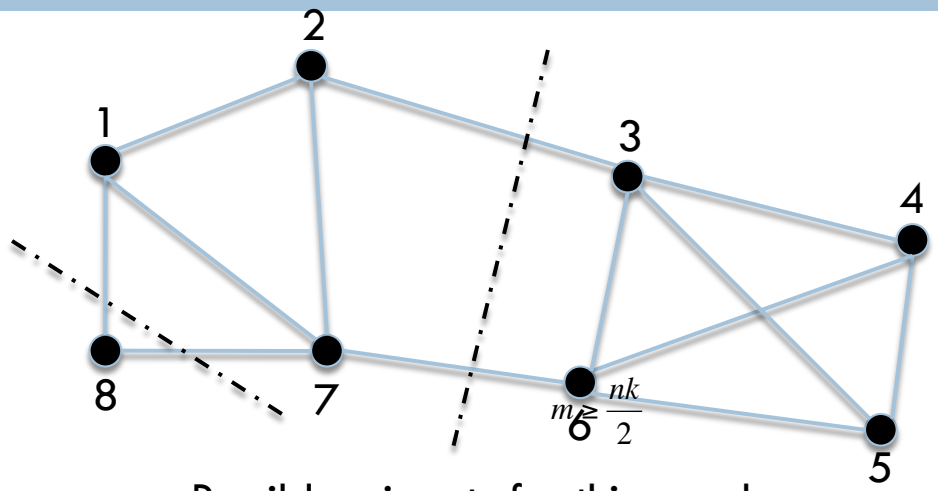
$$\sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{2}{k} \le \sum_{i=1}^{n-1}\sum_{k=1}^{n}\frac{2}{k}$$

$$\le 2n\sum_{k=1}^{n}\frac{2}{k}$$ ➔ O(nlogn)

Las Vegas & Monte Carlo

- Las Vegas
  - Always produces the right answer
  - Its running time is a random variable
- Monte Carlo
  - Answer is approximate
  - Running time is bounded by an upper limit

Min cut in graphs: using Monte Carlo



Possible min cuts for this graph

If a graph has n vertices and m edges and optimum min cut is k,
Then minimum degree of the graph ≥ k

$$m \geq \frac{nk}{2}$$

Monte Carlo Algorithm: min cut

1. Take any random edge and collapse the edge
2. Clean up the graph by deleting self loops but keeping the multi loops
3. Repeat until there are only 2 vertices left

The probability that the optimal min cut k edges are not selected for contraction:

$m_1 = m$	edge 1	$(m-k)/m = 1 - 2/n$
$m_2 = (n-1)k/2$	edge 2	$(m-1-k)/(m-1) = 1 - (2k/nk-2)$
.	.	.
.	Edge n-2	$1 - (2/(n-(n-3))) = 1/3$

Prob (min cut is retained) =

$$\left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\ldots\ldots$$

$$=\left(\frac{2}{n(n-1)}\right)$$

- The probability calculated s pretty low, so to boost the result run the algorithm r times and choose the minimum result k*

Prob(k* = k) ≥ 1 – prob that none of the runs get the optimal min set

$$= \quad 1 - \prod_{i=1}^{r}\left(1 - \frac{2}{n(n-1)}\right) \quad = \quad 1 - \left(1 - \frac{2}{n^2}\right)^r$$

If $r = n^2/2$ prob(k* = k) = $1 - \left(1 - \frac{2}{n^2}\right)^{\frac{n^2}{2}} = 1 - 1/e$

Converting Monte carlo to Las vegas:

M can solve the decision problem with prob α

Running time of M = T(n)

Running time of verification = U(n)

Las Vegas : Min cut

Las Vegas Algo L:

Repeat

- Run M
- Verify answer

Until answer is correct

$$E[T'] = \sum_{i=1}^{\infty} (1-\alpha)^{i-1} \alpha i [T(n) + U(n)]$$
$$= \frac{T(n) + U(n)}{\alpha}$$

P(L ran for exactly i iterations) = $(1-\alpha)^{i-1}\ \alpha$

Time taken for i iterations = i(T(n) + U(n))

Expected running time of L :

$$E[T'] = \sum_{i=1}^{\infty} (1-\alpha)^{i-1} \alpha i [T(n) + U(n)]$$

$$= \frac{T(n) + U(n)}{\alpha}$$

Max 3-SAT Problem

- Decision Problem: trying to find the maximum no of clauses that satisfy
- Algorithm:
  - Uniformly assign random boolean values to

  $x1, x_2, \ldots, x_n$
- If there are m clauses: $C_1, C_2, \ldots, C_m$
- Prob(C_i is satisfied) = 7/8
- Expected no of satisfied clauses = 7m/8
- Two observations:
  - For any max 3 SAT problem with n variables and m clauses

  There will be an assignment such that at least 7m/8 clauses are satisfied
  - For any max 3 SAT problem with at most 7 clauses

  There will be an assignment such that all clauses are satisfied