GA Endterm Notes

Flows and Cuts:
Lecture 13

1. **Maximum Flow**
   Problem: given a (directed) graph G = (V, E) with edge capacities (> 0), source u, sink v, find the max possible “rate” at which one can send “information” from u to v.

   Solution:
   - Find shortest path from u to v and send as much flow as possible.
   - Update capacities on edge (i.e. keep only remaining capacities)
   - Repeat above steps again.
   - This approach depends on path chosen, In case of mistake back push the flow.

2. **Min cut problem**
   Problem: given a (directed) graph G = (V, E) with edge costs (> 0), source u, sink v, find the min possible set of edges to “cut” so that there’s no path from u —> v
   Solution:
   - Cut the edges such that there is no path form u to v.

3. **Flows and Cuts**
   Theorem (easy): G = (V, E) be a weighted directed graph, and u, v be vertices. Let “F” be any flow, interpreting wts as capacities. Let “C” be any cut, interpreting wts as costs. Then F <= C.

   Max Flow = Min cut

Lecture 14:
Randomization:

1. **Finding a frequent element in an array**
   Problem: given an (unsorted) array A[0], A[1], …, A[n-1], and the promise that at least n/3 of the A[i] are 0, find one index i s.t. A[i]=0
- Solution:-
   - **Naïve approach:** Iterate through all the elements and return the first index where A[i]=0 -> O(n)
   - **Randomness(Las Vegas Algorithm):**
     - pick r random index and check if A[r] = 0 -> O(r)
     - Probability of failure = (2/3)^r
     - Probability of Success = 1- (2/3)^r = (1/3)^r

2. **Checking identities**
   Problem: If we have two polynomial P(x) and q(x). Is p(x)==q(x)?
   Solution:
   - **Naïve approach:** Expand P(x) and see if it matches to q(x).
   - **Randomness:**
     - Pick a random integer and check if p(x)==q(x) for that integer.
     - Problem is we can't say if p(x)==q(x) if for single x, p(x)=q(x).
     - Hence, we have r(x) = p(x) - q(x) and r(x) has at most less than equal (max degree of p(x)/q(x)) roots.
     - If we pick x in range [0,2d] where d is max degree of p(x)/q(x), then probability that there are equal is <= 1/2

3. **Primality**
   Problem: given an integer X = a1a2 … an, find if X is prime
   - Taking sqrt of x and dividing number by 1 to sqrt x takes O(2^(n/2)).
   - because an -digit number in binary is roughly of the order 2^n and its square root is 2^(n/2).

4. **Perfect matching**
   Problem: given a bipartite graph G, find if it has a “perfect matching. Determine if it is possible to ``pair up'' all the vertices of U and V and such that
   (a) every element of U is paired with precisely one element of V,
   (b) is paired with only if {i,j} is an edge (in E).

   Solution:
   - Firstly, U should be equal to V.
   - We create a Matrix of U*V (n*n).
   - **Randomness**:
     - Put random values in M where there is edge and 0 if there is no edge.
     - If determinant of M != 0 => perfect matching else not ??
     - If we set the random integer in range [0,2n] then probability that Determinant !=0 (perfect matching) is >= 1/2

   Notes:
   1. Using randomness, two runs of same algorithm doesn't guarantee same output.
   2. The algorithm need not always output right answer
   3. Increase the running time => decrease in failure probability.

Lecture 15:
Expected Running Times
- **Las Vegas Algorithm:**
  - Pick random element till you find the required element.
  - Always Succeed, Never fails but takes infinite time
  - High running time -> higher probability of success

- Expected Value = Integration of x*P(x)dx , where p(x) is probability density function

- **Law of conditional Expectation** = P(x) . E(X|F) +(1-p(x))*E(X|Fbar)

- **Expected Number of tosses of a fair coin before seeing heads?**
  alpha=1/2(1)+1/2(1+alpha)
  Hence, alpha = 2

- **Quick Sort:**
  Problem: given unsorted array A[0, …, n-1], sort it.
  Solution:
  - Taka a random index i s.t A[i] = pivot.
  - Array B = all element < A[i]
  - Array C = all element > A[i]
  - Recursively sort B and C.
  - concatenate. (Sorted B + A[i] + Sorted C)

  Good pivot: median -> O(nlogn)
  Bad Pivot: corner element -> O(n^2)

  Now question arises is what should be a running time in general?
  Running time depends on random choices of pivot => random variable.

  E[X] = P(Pivot is smallest element) * E[X|Pivot is smallest element] + P(Pivot is 2nd smallest element) * E[X|Pivot is 2nd sm allest element] + … + O(n)
  T(n) = 1/n * T(n-1) + 1/n * [T(n-2)+T(1)]+ … + O(n) …. Recursive sub-problem
  T(n) = 1/n [T(i) + T(n-i-1)] + O(n)

Lecture 16/17:
Ball and Bins

- **Markov's inequality**
  - let X be a non-negative random variable with expectation E[X]. Then prob[X >= t*E[X]] <= 1/t.
  - prob[X >= t*E[X]] <= 1/t => prob[X <= t*E[X]] >= 1-1/t.

- **Ball and Bins:**
  Problem: suppose we have n balls and m bins. Imagine throwing the balls into bins, independently and uniformly at random.
  1. What is the expected size of each bin?
  2. Suppose n = m; What is the expected number of bins with exactly 4 balls?
  3. Suppose n = m; What is the probability that there exists a bin with (log n) balls?

  Solution:
  1. Let B_i be the number of ball in bin i.
     B_i = n - sum i!=j B_j
     E[B_i] = n - sum i!=j E[B_j]
     sum j E[B_j] = n
     Using symmetry, all E[B_j] values are equal.
     [B1]+E[B2]+…+E[Bm] = n
     m * E[B]=n
     E[B] = n/m

     Alternative Solution,
     Let B be the number of ball in bin 1.
     X1 = 1 if ball 1 goes into bin 1, 0 otherwise
     X2 = 1 if ball 2 goes into bin 1, 0 otherwise
     and so on…

     B = X1+X2+…+Xn
     E[Xi] = 0.P(Xi=0) + 1P(Xi=1)
     = 0 + 1/m
     B = 1/m * n
     E[B] = n/m

  2. Let B be the number of bin with exact 4 balls.
     X1 = 1 if bin i has 4 balls, 0 otherwise
     X2 = 1 if bin i has 4 balls, 0 otherwise
     and so on…

     B = X1+X2+…+Xm
     E[Xi] = 0.P(Xi=0) + 1P(Xi=1)
     = 0 + nC4 * (1/n)^4 * (n-1/n)^(n-4) [As n=m]
     = e/24

     B = E[Xi] * n
     E[B] = E[Xi] * n

     General form = nCk * (1/m)^k * (1-(1/m))^(n-k)

  3.

- **Linearity of Expectation:**
  E[x + y] = E[x] + E[y]

$E[x*y] = E[x] * E[y]$, when x and y are independent

- **Union bound:**
  P[E1 or E2 or E3 or …En] <= P[E1] + P[E2] + P[E3] + … + P[En]
  P[E1 or E2 or E3 or …En] == P[E1] + P[E2] + P[E3] + … + P[En] , if P[Ei and Ej] = 0 for all i,j **(disjoint sets)**

Lecture 18:
Sampling

Average of elements in array
Problem: let A be an array with n elements, each in interval [-1,1]. Find the average of all elements.

Solution:
sample k of the elements (with replacement), find their "empirical average" (sum/k).

Chebyshev's inequality: prob[error >t\sqrt$k$]<= 1/t^2

Lecture 19:
Sampling

Lecture 20/21/24:
Optimization:

3 problems:
- What are variables? - should be binary {0,1}
- Constraints?
- Objective?

- **Matching in bipartite graph:**

- **Set Cover:**

U = People, V = Skills. The goal in set cover is to pick the smallest possible subset of S of U such that all the skills on t he right are "covered".
Variable:
- Xu -> defines if Xu is chosen or not.

Constraints:
- For each skill j there should be at least 1 person.
- If T defines set of people having skill j then Sum (u belong to T), Xu>=1

Objective:
- Minimize S. S = Sum (u belongs to U) Xu

- **Minimum Spanning Tree:**

Variable:
- Xe -> defines if Xe is chosen or not.
- We -> weight across edge Xe

Constraints:
- (n-1) edges are chosen in total. Sum (e belongs to E) Xe = n-1
- Xe = {0,1}
- For any cycle in graph, Sum of edges should be less that vertex covering the cycle - 1.

Objective:
- Minimize S. S = Sum (e belongs to E) We * Xe

Problem: can we search for an element x in a sorted, n-element array in time < log n?
Solution:

Answer should be Decision: Yes/No

Lecture 25:
NP Problems:

Problem: Most Puzzle
- Witness: Solution S to Puzzle
- Verifier: a procedure that checks validity of S.

Problem: Travelling Salesman Problem
- Witness: Order in which to visit vertices
- Verifier: Check if all vertices are visited at least once and distance should be less than allowed max distance.

Problem: Primality Testing
- Witness: Factors of N
- Verifier: If Factor contains other than 1 and number