

1 Topics

- Data Structures (PS1)
- Dynamic Programming/Shortest Paths
 - Relaxation Algorithms
 - Dijkstra's Algorithm
 - Bellman-Ford
 - Applications
 - Price Functions
- Max Flow - Min Cut
 - Augmenting Paths Algorithm and Proof
 - Residual Graphs G_f
 - Blocking Flow Definition
Always increases the distance from s to t after an augmentation.
 - Unit Capacity Graphs
Lecture notes and Problem set
- Fingerprint
 - Polynomial Equivalence
Bounds on the equivalence of two polynomials with random inputs
 - Randomized Pattern Matching
 - File Equivalence
- Chernoff Bounds / Balls in Bins
 - Server Loads
 - Path Scheduling
- Linear Programming
 - Write Linear Programs for various problems (ie: shortest paths, flow problems, generalizations, etc...)
 - Simplex Algorithm (walk from vertex to vertex, termination)
 - Ellipsoid Algorithm
Held-Karp bound: For all subsets S , assign $x_e \leq 1$ and $\sum_{e=S \times S'} x_e > 2$. Then minimize $\sum c_e x_e$. There are an exponential number of constraints. Min cut is used as the separation algorithm
Ellipsoid algorithm doesn't need the constraints, just need to produce them as you use the algorithm. Works with a separation oracle—with a set of constraints, we can find a violated constraint.
- Codes
 - Shannon's Result (Shannon Capacity of a Binary Symmetric Channel)
 - Hamming's "type" Results (Minimum distance between codewords)
Gilbert-Vorshonov Result: Random Linear Codes have minimum distance ?

- Approximation Algorithms
 - Vertex Cover (bipartite matchings)
 - Set Cover (greedy algorithm)
 - Traveling Salesperson (TSP)
- Nearest Neighbors (not very important)
 - Curse of dimensionality (why is it hard in high dimensions?)
 - Thursday’s lecture. COME!
- Randomized Algorithms
 - Randomized minimum cut
 - Randomized minimum spanning tree
- Randomized Tools
 - Counting
 - Applying Chernoff Bounds
 - Expectation (Linearity of Expectation for MST/MinCut)

2 Questions/Problems

1. Is Dijkstra’s linear time on a weighted tree.
No, it is $V \log V + E$. On a tree, Dijkstra’s essentially sorts the weights of the tree. s
2. Design an algorithm that runs faster than Bellman-Ford with negative edges
We’ll still need n phases, so there is nothing quicker than BF. However we can do better if we know the number of negative edges (or if there is just one)
3. What’s the max load of putting n balls in n bins with high probability.
 $\log n$ is a decent answer. $\log \log n$ is even better. $\Theta(\frac{\log n}{\log \log n})$ was proven in the problem set. Here’s how to prove an upper bound:

$$\begin{aligned} \Pr[\text{bin } i \text{ has } = k \text{ balls}] &= \binom{n}{k} \left(\frac{1}{k}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \\ &\leq \binom{n}{k} \left(\frac{1}{n}\right)^k \\ &\leq \left(\frac{e}{k}\right)^k \end{aligned}$$

$$\begin{aligned} \Pr[\text{bin } i \text{ has } \geq k \text{ balls}] &\leq \left(\frac{e}{i}\right)^k && i \geq k \\ &\leq c \left(\frac{e}{k}\right)^k \end{aligned}$$

Or use Chernoff bounds and define the random variables:

$$X = \sum x_i$$
$$x_i = \begin{cases} 1 & \text{if ball } i \text{ falls in bin } j \\ 0 & \text{otherwise} \end{cases}$$
$$\mu = E[X] = 1$$