Advanced Algorithms

Lecture 14: Randomness in algorithm design

Announcements

- **Mid-term grades out**
- HW 3 due Wednesday (tomorrow)

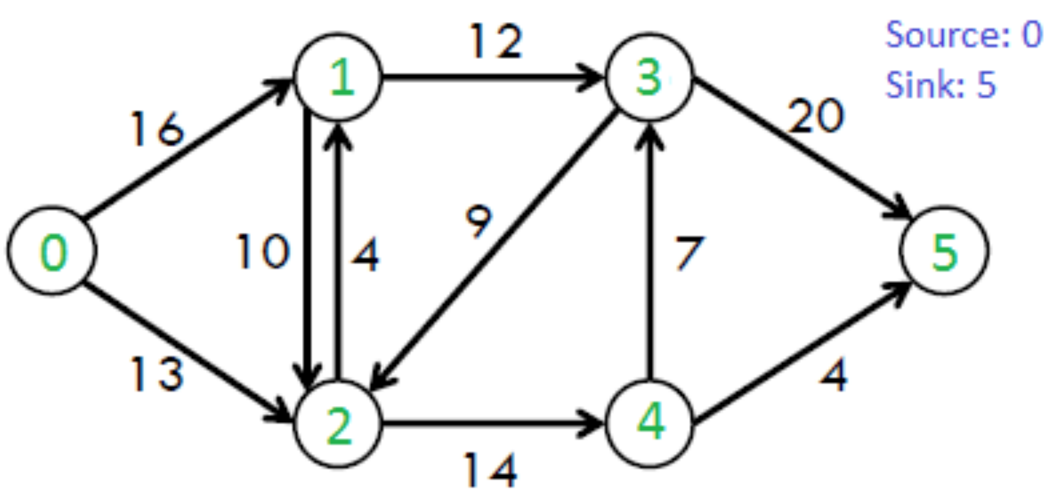
Last two weeks

- Basic graph algorithms
  - Dijkstra's algorithm (O(m+n) log n) time — imitation of BFS
  - DP based, "Bellman-Ford" algorithm — O(n (m+n)) time
  - "Definitions" of flows and cuts in graphs

Maximum flow

communication networks, shipping goods, …

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.



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*

Blowing bridges…

- Undirected graphs
- Image segmentation

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.

Comments

- Max-flow min-cut theorem

- Many applications — e.g., no bottleneck => many edge disjoint paths

- Algorithms for cut == algorithms for flow

Today

Can randomness help in algorithm design?

Toy problem

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

- Generalization of HW problem

Randomized procedure

Key trade-off

- Higher running time, higher probability of success
- Note: don’t even read entire input!

“Las Vegas” algorithm

- While not found: pick random index i and check if A[i]=0

Expected Running Time

(similar to tossing until seeing heads)
Running time is a random variable

Example 2 – checking identities

$$p(x) = (x-7)(x-3)(x-1)(x+2)(2\,x+5)$$
$$q(x) = 2\,x^5 - 13\,x^4 - 21\,x^3 + 127\,x^2 + 121\,x - 210$$

- What if we simply plug in a random integer *x* in interval [1,20]?

One variable identities

General theorem: Schwartz-Zipfel Lemma

Example 3 – primality

Problem: given an integer $X = a_1 a_2 \ldots a_n$, find if X is prime

- Classic problem in math/CS
- Can an algorithm run in time poly(n)?
- Miller-Rabin test

Example 4 – perfect matching

Problem: given a bipartite graph G, find if it has a “perfect matching”

- Claim: this reduces to identity testing!

Perfect matching

Examples so far

- Finding hay in a hay stack
- Trade-off between running time and success probability
- (Fairly general) — “boosting”

Randomized algorithms overview

- Data is given, algorithm is randomized (unlike sampling/“ML” analyses)
- Usually concerned about **expected behavior**, behavior “with high probability”

Next few lectures: **general ideas, applications, analysis…**