Advanced Algorithms – Randomised Algorithms

Graph Algoritms

Georgia Kaouri, NTUA

Contents

- All-Pairs Shortest Paths
  - All-pairs distances
  - Boolean product witness matrix
  - Determine Shortest Paths
- The Min-Cut Problem
  - Contraction Algorithm-Fast-Cut
  - Karger's Reliability Theorem
- Minimum Spanning Trees
  - Borůvka's Algorithm
  - Linear Time MST Algorithm

All-pairs Shortest Paths

- Def: G(V, E) undirected, connected graph, |V|=n, |E|=m
- Adjacency matrix A -> Distance matrix D
- All-Pairs Shortest Paths (APSP)
  - $O(n^3)$, $O(mn+n^2 \log n)$
  - $\Omega(n^2)$
- All-Pairs Distances (APD)

- What do A^2, A^3, …, A^n express?

Solving APD (deterministically)

- Z <- A^2
- Compute matrix A′ such that $A'_{ij}=1$ if and only if $i \neq j$ and ($A_{ij}=1$ or $Z_{ij}>0$)
- If $A'_{ij}=1$ for all $i \neq j$ then return $D = 2A'-A$.
- Recursively compute the APD matrix D′ for the graph G′ with adjacency matrix A′.
- S <- AD′
- Return matrix D with $D_{ij}=2D'_{ij}$ if $S_{ij} \geq D'_{ij}Z_{ii}$, otherwise $D_{ij}=2D'_{ij}-1$.

Analysis I

- Compute G′(V, E′), the "square" of G, i≠j (i, j)∈E′ iff there exists path of length 1 or 2 between i, j in G.
- Easy: $Z=A^2$, A determine A′ in $O(n^2)$
- Observation: G′ complete iff G has diameter 2 -> D=2A′-A in $O(n^2)$
- Idea: Compute recursively D from D′
- What is the relationship between D and D′?

Analysis II

- PARITY Lemma: For every pair i, j Є V
  - If D_{ij} is even then $D_{ij}=2D'_{ij}$
  - If D_{ij} is odd then $D_{ij}=2D'_{ij}-1$
- Lemma: For every pair i, j Є V
  - For any $k \in \Gamma(i)$: $D_{ij} \le D_{ik}+D_{kj}$ -> $D_{ij}-1 \le D_{kj}$ and $D_{kj} \le D_{ki}+D_{ij}$ -> $D_{kj} \le 1+D_{ij}$
  - There exists a $k \in \Gamma(i)$ s.t. $D_{jk}=D_{ij}-1$
- Lemma: For every pair i, j Є V
  - If D_{ij} is even then $D'_{kj} \ge D'_{ij}$ for every $k \in \Gamma(i)$ in G
  - If D_{ij} is odd then $D'_{kj} \le D'_{ij}$ for every $k \in \Gamma(i)$ in G

    Moreover there exists a k $k \in \Gamma(i)$ s.t. $D'_{kj} < D'_{ij}$

Analysis III

- Summing over all vertices in Γ(i) the second part of the inequality is multiplied by d(i)

- The APD algorithm solves APD in O(MM(n)logn)
  - If G has diameter δ G' has δ/2.
  - $T(n, \delta)=2MM(n) + T(n,\delta/2) + O(n^2)$
  - δ=1 G is complete
  - δ=2 $T(n, \delta)=MM(n) + O(n^2)$

Boolean Product Witness Matrix (BPWM)

- Suppose A and B are $n \times n$ boolean matrices and $P=AB$ is their product under Boolean matrix multiplication.
- A **witness** for P_{ij} is an index $k \in \{1,\ldots,n\}$ such that $A_{ik}=B_{kj}=1$. Observe that $P_{ij}=1$ if and only if it has some witness k.
- BPWM for P is a matrix W, where $W_{ij} \in \{1,2,\ldots,n\}$ contains a witness iff $P_{ij}=1$
- If A=B the adjacency matrix $P_{ij}=1$ iff there exists a path of length 2 in G, therefore a witness k is the intermediate vertex.

Analysis I

- Simple case: unique witness
  - multiply column k by k
  - read off witness identity
- Reduction: If r columns have witnesses, choose each column with probability p
- R≤{1,2,…,n} random set of cardinality r
- What is the probability to find exactly 1 witness?
  - urn(w white, n-w black). Choose r balls at random, with $n/2 \leq wr \leq n$. Then

    Pr{exactly one ball is chosen}≥1/2e
- Try all values of r!!!???

Analysis II

- Represent R as a vector with $R_k=1$ iff k∈R
- AR: $A^R_{ik}=kR_kA_{ik}$
- So we will choose each column with probability 2^{-k} s.t $1\leq 2^k\leq n$ O(logn)tries
- The probability of finding exactly one witness is: $r\ 2^{-k}\ (1-2^{-k})^{r-1} \geq (1/2)(1/e^2)$
- Repeat logn times
- Las Vegas algorithm

All-pairs Shortest Paths

- Compute distances with APD
- For each i,j find a node $k \in \Gamma(i)$ on the shortest path (have distance one less)
- Keep an array R saying that the distance is -1
- Compute the boolean witness product RA
- How many matrix multiplications should we do?

Analysis

- Recall that for every pair i, j Є V
  - For any $k \in \Gamma(i)$: $D_{ij}-1 \leq D_{kj} \leq D_{ij}+1$
  - For any $k \in \Gamma(i)$ with $A_{ik}=1$ and $D_{kj}=D_{ij}$ (mod3) is valid candidate foa being a successor of i on the shortest path to j.

Algorithm APSP

- Compute the distance matrix D=APD(A).
- for s={0,1,2} do
  - Compute 0-1 matrix $D^{(s)}$ with $D_{kj}^{(s)}$=1 iff D_{kj}+1 = s (mod 3)
  - Compute the witness matrix $W^{(s)}$=BPWM(A,$D^{(s)}$).
- Compute successor matrix S for G.

Min-Cut Problem

- Input: an undirected, connected multigraph $G = (V,E)$
- Output: A cut (V_1, V_2 where $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$) such that number of edges between V_1 and V_2 is the fewest possible.
- Contraction Algorithm
- FastCut

Network Reliability

- Given a network of n vertices, m edges, each edge has probability of failure p_e
- What is the probability that the surviving network is disconnected?
- The problem is in #P.
- The algorithm works on reliable graphs and computes the probability for the network to be reliable.

More Lemmas

- Of all graphs with min cut c, the least reliable is the cycle on n nodes with c/2 edges between adjacent nodes.
- If each edge of a graph with min cut c is removed with probability p, the probability that the network fails is at least p^c and at most n^2p^c.
  - the c edges in some min cut fail with probability p^c
  - for the above graphs, the probability that 2 sets of c/2 edges fail is p^c and there are (n choose 2) pairs of groups of edges
- In a graph with min cut c, there are at most $n^{2\alpha}$ cuts with less than αc cycles.

Reliability Theorem

- Suppose a graph has min cut c, s-t cut u and each edge fails with probability p, $p^c < n^{-(2+\varepsilon)}$ for some ε. Then the probability that the network becomes disconnected is $O(n^{-\varepsilon}(1+1/\varepsilon))$, and the probability that s and t become disconnected is $O(n^{-u/c}(1+1/\varepsilon))$.
- Let $r=2^n-2$ the #cuts, $c_1,c_2,\ldots,c_r$ their values, with $c=c_1\leq c_2\leq\ldots\leq c_r$ and $p_k=p^{c_k}$ the probability that all edges in k-th cut fail.
  – The probability that G disconnects is Σp_k. Let's bound it from above!

- Consider the n^2 smallest cuts. Each one is larger than c, therefore $p_k \leq n^{-(2+\varepsilon)}$:

  $\Sigma_{k\leq n^2} p_k \leq n^2 n^{-(2+\varepsilon)} = n^{-\varepsilon}$.

- Consider the remaining larger cuts. From a previous theorem, there are at most $n^{2\alpha}$ cuts of value less than αc, i.e. $c_{n^{2\alpha}} \geq \alpha c$. For $k = n^{2\alpha}$:

  $c_k \geq [\ln k/2\ln(2n)]\ c$

  And $p_k \leq (p^c)$^{lnk/2ln(2n)}$= k^{-(1+\varepsilon/2)}$.

- Therefore $\Sigma_{k> n^2} p_k \leq \Sigma\ \Sigma_{k> n^2} k^{-(1+\varepsilon/2)} \approx O(n^{-\varepsilon}/\varepsilon)$

An Approximation Algorithm

- E_i is the set of edges in the i-th small cut.
- Assign a boolean variable x_e to each edge e, with x_e true if edge e fails. x_e are independent and true with probability p.
- i-th cut fails: $F_i = \cap_{e<E_i} x_e$.
- Some small cut fails: $F = \cup_i F_i$.
- We wish to know the probability that F is true!
- F is in DNF with $n^{2\alpha}$ clauses and at least c variables per clause.
- Karp, Luby and Madras {KLM89} estimated the truth probability in this formula, thus the failure probability.

Minimum Spanning Trees

- G(V,E) connected graph with edge weights w:E -> R with n vertices and m edges.
- A spanning tree is an acyclic graph of G that includes every vertex in G and is connected.
- We wish to compute the minimum spanning tree (MST).

Borůvka's Phase

- Contract simultaneously the min weight edges incident on each of the vertices in G.
- Implementation:
  - mark the edges to be contracted
  - determine the "new" connected components
  - replace the "new" connected components with a single vertex
  - eliminate self loops and multiple edges created
- Runs in O(mlogn) time

Heavy and Light Edges

- Fix a forest on G and any pair of vertices $u,v \in V$.
- $w_F(u,v)$ denotes the max weight of any edge on the path $P(u,v)$ if it exists, ∞ otherwise
- Edge (u,v) is F-heavy if $w(u,v) > w_F(u,v)$

  F-light if $w(u,v) \leq w_F(u,v)$
- If an edge is F-heavy, it does not lie in the MST.
- An F-light edge can be used to improve the MST.
- A verification algorithm for MST takes a candidate MST, checks that only F-light edges are used and accepts if they are, returns the F-light edges if they aren't.

Random Graphs

- A random graph G(p) is obtained by graph G by including independantly each edge of G in G(p) with probability p.
- G(p) has n vertices and mp expected edges.
- We expect that very few edges in G are F-light.
- Random variable X has the negative binomial distribution with parameters n and p, if it corresponds to the number of independent trials required for n successes when each trial has probability of success p.
- X stochastically dominates Y if for all $z \in R$ $Pr[X>z] \geq Pr[Y>z]$.
- If X stochastically dominates Y then $E[X] \geq E[Y]$.

F-light edges are few…

- Let F be the min spanning forest in G(p). Then the number of F-light edges in G is stochastically dominated by a random variable X that has the negative binomial distribution with parameters n and p. The expected number of edges in G is at most n/p.
  - $e_1, e_2, \ldots, e_m$ the edges in increasing weight
  - Construct the MSForest F online while choosing edges for G(p).
  - At step i if edge e_i is chosen for G(p), it is a candidate for F.
  - Edge e_i is added to F iff it connects previously disconnected components.
  - Note that
    - Whether an edge in F if F-light depends on the coin
    - edges are never removed
    - e_i is F-light at the end iff it is F-light at the beginning of step i.

- k-1 edges have been added to F. Phase k begins and ends when k edges belong to F.
- An F-light edge is added in F only if the coin says so, but phase k ends when such an edge is added.
- Therefore, during this phase we have some F-light edges that are chosen with probability p -> F-light edges have the negative binomial distribution with parameter p.
- In total F grows from 0 to s: continue to flip the coin until n HEADS have appeared. The random variable that expresses the total number of coin flips has the negative binomial distribution with parameters n, p.
- Therefore, the expected number of F-light edges is bounded by n/p.

Linear-Time MST Algorithm

- Use 3 applications of Borůvka's Phase to compute G_1 with at most n/8 vertices and let C be the set of contracted edges. If G is empty return F=C.
- Let $G_2=G_1(p)$, with p=1/2
- Recursively apply MST, compute minimum spanning forest F_2 for graph G_2.
- Use a verification algorithm to identify F_2-heavy edges in G_1 and delete them to obtain graph G_3.
- Recursively apply MST to compute the minimum spanning forest F_3 for graph G_3.
- return forest $F=C\cup F_3$.

The expected running time of MST is O(n+m)

- T(n,m)=T(n/8,m/2)+T(n/8,n/4)+c(n+m):
  - Borůvka's Phase O(n+m)
  - G2 has n/8 vertices and m/2 expected edges runs in O(n+m)
  - find min spanning forest in G2 in expected time T(n/8,m/2)
  - verification takes O(n+m) and produces G3 with at most n/8 vertices and expected n/4 edges
  - find the min spanning forest of G3 has expected cost T(n/8,n/4)
  - Return the final forest in O(n)

- Thank you…