CS 170 DIS 11

Released on 2018-11-13

1 Local Search for Max Cut

Sometimes, local search algorithms can give good approximations to NP-hard problems. In the Max-Cut problem, we have a graph $G(V, E)$ and want to find a cut (S, T) with as many edges crossing as possible. One local search algorithm is as follows: Start with any cut, and while some vertex v in S has more neighbors in S than T, we move v from S to T (we do the same for any vertex v in T with more neighbors in T than S). Note that any time we move a vertex across the cut, the number of edges crossing the cut increases.

(a) Give an upper bound on the number of iterations this algorithm can run for (i.e. the total number of times we move a vertex).

(b) Show that when all vertices have more neighbors on the opposite side of the cut, at least half the edges in the graph cross the cut.

2 Multiway Cut

In the multiway cut problem, we are given a graph $G(V, E)$ with k special vertices $s_1, s_2 \ldots s_k$. Our goal is to find the smallest set of edges F which when removed from the graph disconnect the graph into at least k components where each s_i is in a different component. When $k = 2$, this is exactly the min s-t cut problem, but if $k \geq 3$ the problem becomes NP-hard.

Consider the following algorithm: Let F_i be the set of edges in the minimum cut with s_i one one side and all other special vertices on the other side. Output F, the union of all F_i. Note that this is a multiway cut because removing F_i from G isolates s_i in its own component.

(a) Explain how each F_i can be found in polynomial time.

(b) Let F^* be the smallest multiway cut. Consider the components that removing F^* disconnects G into, and let C_i be the vertices in the component with s_i. Let F_i^* be the set of edges in F^* with exactly one endpoint in C_i. How many different F_i^* does each edge in F^* appear in? How do the size of F_i and F_i^* compare?

(c) Using your answer to the previous part, show that $|F| \leq 2|F^*|$. (Challenge: How could you modify this algorithm to output F such that $|F| \leq (2 - \frac{2}{k})|F^*|$?)

(As an aside, consider the minimum k-cut problem, where we want to find the smallest set of edges F whose removal disconnects the graph into at least k components. The following greedy algorithm for minimum k-cut gets a $(2 - \frac{2}{k})$-approximation: Initialize F to the empty set. While $G(V, E - F)$ has less than k components, find the minimum cut within each component of $G(V, E - F)$, and add the edges in the smallest of these cuts to F. Showing this is a $(2 - \frac{2}{k})$-approximation is fairly difficult.)

3 Fast Modular Exponentiation

Give a polynomial time algorithm for computing $a^{b^c} \mod p$ for prime p and integers a, b, and c.

4 Fermat's Little Theorem as a Primality Test

Recall that Fermat's Little Theorem states the following:

"For a prime p and a coprime with p, $a^{p-1} \equiv 1 \pmod p$."

Assume for a general (not necessarily prime) p, we want to determine if p is prime. It may be tempting to try to use Fermat's Little Theorem as a test for primality. That is, pick some random a and compute $a^{p-1} \pmod p$. If this is equal to 1, return that p is prime, else return that it is composite. In this question we will investigate how effective this method actually is.

(a) Suppose we wanted to test if 15 was prime. What is a choice of a that would trick us into thinking it is prime? What is a choice of a that would lead us to the correct answer? For choices of a that trick us into believing p is prime, we often say that p is "Fermat pseudoprime" to base a.

(b) Suppose there exists some a in $\{1, \ldots p-1\}$ such that $a^{p-1} \not\equiv 1 \pmod p$, where a is coprime with p. Show that p is not Fermat pseudoprime to at least half the numbers in (mod p). How might we use this to make our algorithm more effective?

(c) Given the improvement from the previous question, why might our algorithm still fail to be a good primality test?