Advanced Algorithms: Exam 2 Review Solutions

Dana Randall Spring 2024

Sarthak Mohanty

Exercise 1

Short-answer questions. No justification required.

(a) Consider Fortune's algorithm to compute the Voronoi diagram of n sites in the plane. What is the maximum number of arcs that any one site can contribute to the beach line?

(b) Which of the following assertions about the Delaunay triangulation (DT) of a set P of n points in the plane are true? Select all that apply.

i. The closest pair of points are connected by an edge in the DT.

ii. The farthest pair of points cannot be connected by an edge in the DT.

iii. Among all triangulations of P, the DT maximizes the minimum angle.

iv. Among all triangulations of P, the DT minimizes the maximum angle.

v. Among all triangulations of P, the DT minimizes the total sum of edge lengths

(c) Recall the MaxExact3SAT problem. The input is a Boolean formula in conjunctive normal form such that every clause is the disjunction of exactly three literals and you wish to output an assignment of the variables that satisfy the maximum number of clauses. True or false: there exists an ϵ-approximation for the MaxExact3SAT problem for any $\epsilon > 0$.

(d) True or false: there exists a 2-approximation for the TSP problem.

Solution

(a) The maximum number of arcs that any one site can contribute to the beach line is n. This occurs when each site (excluding the first) splits an arc originating from the first site.

(b) The correct choices are i and iii.

(c) False. The lecture notes give a categorization of many NP-Complete problems in terms of their approximability. There we can see that MaxSAT is approximable for some but not all ϵ. A proof is given on Wikipedia.

(d) True, a 2-approximation for TSP is given in the notes:

Find a minimum spanning tree T of G. Double every edge of T to obtain a Eulerian graph. Find a Eulerian tour $\mathcal{T}$ on this graph. Output the tour

that visits the vertices of G in the order of their first appearance in T. Let $\mathcal{C}$ be this tour.

Proof of correctness: Note that $|T| \leq \text{OPT}$ because deleting an edge from an optimal tour yields a spanning tree. Moreover, $|\mathcal{T}| = 2 \cdot |T|$. Because of the triangle inequality, $|\mathcal{C}| \leq |\mathcal{T}|$. Hence, $|\mathcal{C}| \leq 2 \cdot \text{OPT}$.

Exercise 2

(Erickson) Suppose you are a TA for a computational geometry class. You're allowing the students to use their own favorite programming language and programming environment, which means you can't compile the code yourself. Thus, instead of submitting the code itself, you asked your students to submit several Voronoi diagrams computed by their code.

After the submission deadline, you realize that you forgot to ask the students to submit the sites that defined each of their Voronoi diagrams! In frustration, you decide to give your students full credit if every diagram they submit is the Voronoi diagram of some point set. But how do you tell?[1]

Describe and analyze an algorithm to determine, given a planar straight-line graph G, whether G is a Voronoi diagram. Your algorithm should either return a finite point set P such that G is the Voronoi diagram of P, or report correctly that no such point set exists. [*Hint:* Consider the case of four sites.]

Exercise 3

(Mount) In class we proved that the Delaunay triangulation of a set of sites in the plane maximizes the minimum angle. (It is the max-min angle triangulation.) Unfortunately, it is not the best triangulation for the following two criteria.

(a) Give an example of a set of point sites in the plane such that the Delaunay triangulation of this set does not minimize the sum of edge lengths, among all possible triangulations. In other words, the Delaunay triangulation is not the minimum-weight triangulation.

[1] I had a similar issue as a TA for 2051.

(b) Give an example of a set of point sites in the plane such that the Delaunay triangulation of this point set does not minimize the maximum angle, among all possible triangulations. In other words, the Delaunay triangulation is not the min-max angle triangulation.

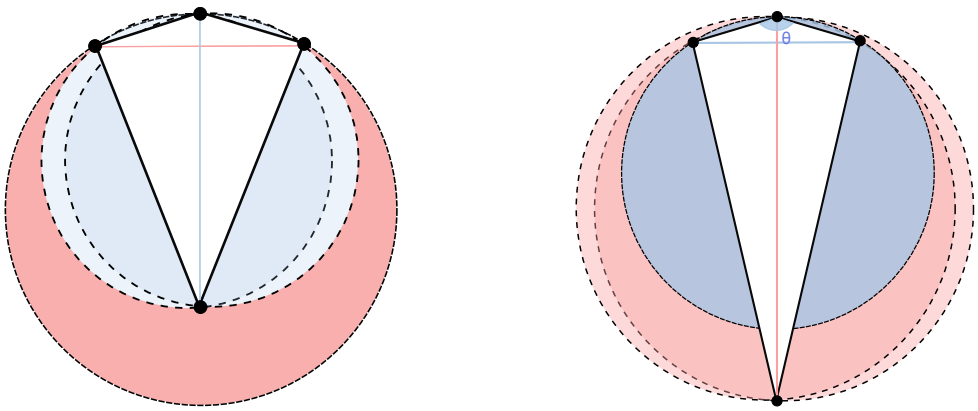
In each case briefly explain your construction. Your example should be in general position (e.g., no four sites should be cocircular).

Hint: In both cases, it is possible build a counterexample consisting of just four points that are nearly co-circular. It suffices to present a single, specific example.

Solution

(a) The example is shown below in Figure 1(a). By flipping the highlighted edge in the Delaunay triangulation, we can create a minimum-weight triangulation that is not Delaunay.

(b) The example is shown below in Figure 1(b). By flipping the highlighted edge in the Delaunay triangulation, we can create a min-max angle triangulation that is not Delaunay.



(a) (b)

Figure 1: Here blue denotes a Delaunay triangulation, and red denotes a non-Delaunay triangulation.

Exercise 4[2]

Suppose you want to compute a Voronoi data structure for a set of pixels drawn from an R-by-R grid. Devise an efficient scheme to support the following three operations.

- `create(R)`: create an empty data structure to handle an R-by-R grid.
- `insert(i,j)`: insert the pixel (i, j) into the data structure, where i and j are integers between 0 and $R - 1$.
- `find(x,y)`: return the inserted pixel which is closest to (x, y) in Euclidean distance, where x and y are integers between 0 and $R - 1$.

The `create` and `insert` operation should take $\mathcal{O}(R^2)$ time; the `find` operation should take $\mathcal{O}(1)$ time. For partial credit, give an algorithm that takes $\mathcal{O}(NR^2)$ time for `insert`, where N is the number of points already inserted into the data structure. (*Hint:* think simple.)

(a) Describe how to implement `create(R)`. Indicate what is being stored.

(b) Describe how to implement `find(i,j)`.

(c) Describe how to implement `insert(i,j)`.

Solution

(a) `create(`R`)`. Use a helper data type `Pixel` to manipulate pixels. Initialize an R-by-R grid of pixels to `null` so that `nearest[`i`][`j`]` is the inserted pixel closest to (i, j).

(b) `find(`i, j`)`. Return `nearest[`i`][`j`]`.

(c) `insert(`x, y`)`. Create a new `Pixel` object p with coordinates (x, y). For each i and j, check whether (i, j) is closer to p than it is to `nearest[`i`][`j`]`. If it is, update `nearest[`i`][`j`]`.

[2]From Princeton COS 226 Final Exam.

Exercise 5

Assess the validity of the following claim: "If there exists an ϵ such that MaxClique is ϵ-approximable, then MaxClique is also $\frac{\epsilon}{4}$-approximable." If true, then give a $\frac{\epsilon}{4}$-appoximation for MaxClique. If false, provide a counterexample or reasoning to demonstrate why the claim does not hold.

Solution

True. Given an ϵ-approximation for MaxClique, we can construct an $\frac{\epsilon}{2}$-appoximation for MaxClique (the construction is given in the notes). We can then repeat this process once more to obtain an $\frac{\epsilon}{4}$-appoximation for MaxClique.

Exercise 6

Prove that Chebyshev's inequality is tight: in other words, find a probability distribution for X and a value a such that

$$\Pr\left(|X - \mathrm{E}\left[X\right]| \geq a\right) = \frac{\mathrm{Var}\left(X\right)}{a^2}.$$

Hint: This random variable will take only three values.

Solution

Let $a = 1$ and X be a random variable such that

$$p_X(x) = \begin{cases} \frac{1}{2} & \text{if } x = -1 \text{ or } x = 1, \\ \frac{1}{2} & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathrm{E}\left[X\right] = 0$ and $\mathrm{Var}\left(X\right) = 1$, so

$$\Pr\left(|X - \mathrm{E}\left[X\right]| \geq a\right) = \Pr\left(|X| \geq 1\right) = 1$$

and

$$\frac{\mathrm{Var}\left(X\right)}{a^2} = 1.$$

Exercise 7

As in the homework, consider the general problem where we have m machines $M_1 \ldots M_m$ and a set of n jobs. Each job j has a processing time t_j. We seek to assign each job to one of the machines so that the loads placed on all machines are as "balanced" as possible. That is, if $A(i)$ are the set of jobs assigned to machine M_i , then the load for machine M_i is total time required.

$$T_i = \sum_{j \in A(i)} t_j$$

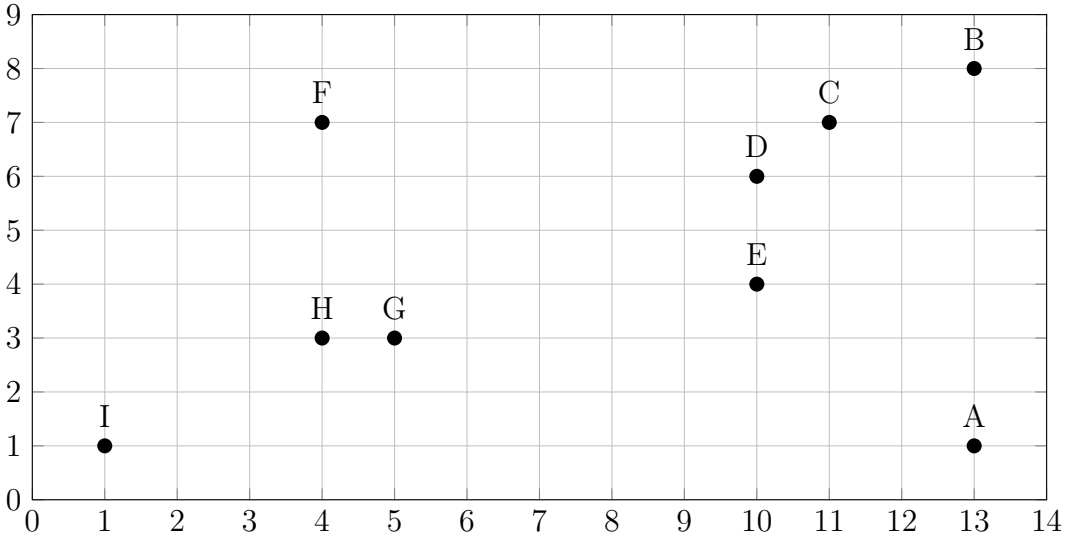
We want to minimize the *makespan* $T = \max_i T_i$.

(a) Describe a set of jobs such that the makespan of the greedy assignment is exactly $(2 - 1/m)$ times the makespan of the optimal assignment, where m is the number of machines.

(b) Describe an efficient algorithm to solve the minimum makespan scheduling problem exactly if every processing time T_i is a power of two.

Exercise 8

Run the Graham scan algorithm (as described in the lecture notes) to compute the convex hull of points below, starting from A. Give the points that appear on the trial hull (after each iteration) in the order that they appear.



Solution

1. A
2. AB
3. ABC
4. ABCD
5. ABCDE
6. ABCDEF
7. ABFG
8. ABFGH
9. ABFHI

Exercise 9

Again consider the following heuristic for constructing a vertex cover of a connected graph G: return the set of non-leaf nodes in any depth-first spanning tree of G. We know (from the homework) this algorithm is a 2-approximation to the minimum vertex cover. Describe an infinite family of graphs for which this heuristic returns a vertex cover of size $2 \cdot OPT$.

Exercise 10

It is a fact that we still have no closed-form expression for the tail of the Normal distribution, which must be computed by numerically evaluating the integral. Let $X \sim \mathcal{N}(0, 1)$. Your goal is to produce upper bounds on $P(X \geq a)$, where $a > 0$.

(a) Use Markov's inequality to bound $\Pr(X \geq a)$. Note: This is not as trivial as it might seem because X is not non-negative. It will help to observe that:

$$\Pr(X \geq a) = \Pr(X \geq a \mid X > 0) \cdot \Pr(X > 0).$$

Now define the non-negative r.v. $Y = [X \mid X > 0]$ and note that $\Pr(X \geq a \mid X > 0) = \Pr(Y \geq a)$.

(b) Use Chebyshev's inequality to bound $\Pr(X \geq a)$.

(c) *(Optional)* Derive the Chernoff bound for $\Pr(X \geq a)$.

Solution

(a) Define the non-negative r.v. $Y = [X \mid X > 0]$. Then

$$\begin{aligned}\Pr(X \geq a) &= \Pr(X \geq a \mid X > 0) \cdot \Pr(X > 0) \\ &= \Pr(Y \geq a) \cdot \Pr(X > 0) \\ &\leq \frac{\mathrm{E}[Y]}{a} \cdot \Pr(X > 0)\end{aligned}$$

(b) Chebyshev's inequality states that

$$\Pr(|X - \mathrm{E}[X]| \geq a) \leq \frac{\mathrm{Var}(X)}{a^2}$$

In our case, $X \sim \mathcal{N}(0, 1)$. Therefore, Chebyshev's inequality becomes:

$$\Pr(|X| \geq a) \leq \frac{1}{a^2}$$

Since X is symmetric about its mean (zero), we can write:

$$\Pr(X \geq a) = \frac{1}{2} \Pr(|X| \geq a)$$

Therefore, applying Chebyshev's inequality gives:

$$\Pr(X \geq a) = \frac{1}{2} \Pr(|X| \geq a) \leq \frac{1}{2} \cdot \frac{1}{a^2} = \frac{1}{2a^2}$$

(c) See here for the solution

Exercise 11

Previously in this class, we discussed the **coupon collector** problem (do you remember when?) As a reminder: there are n distinct types of coupons you would like to collect. Each day you are sent a random coupon from among the n types. Let X denote the number of days needed to collect all n distinct coupons, given that coupons are chosen randomly with replacement. The following (famous) identity[3] is useful in answering some of the questions below:

$$\sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6}.$$

[3] known as the Basel problem.

(a) What is $\mathrm{E}[X]$?[4] What does this approach for high n? Write your answer using $\mathcal{O}(\cdot)$.

(b) Derive $\mathrm{Var}(X)$. What does this approach for high n? Write your answer using $\mathcal{O}(\cdot)$.

(c) Derive an asymptotic upper bound on $\Pr(X \geq 2n\ln(n))$ for large n using Markov's inequality.

(d) Derive an asymptotic upper bound on $\Pr(X \geq 2n\ln(n))$ for large n using Chebyshev's inequality. Express your answer using $\mathcal{O}(\cdot)$.

Note: For $\mathrm{E}[X]$ in (c) and (d), use the asymptotic mean from part (a).

Solution

(a) Let X_k denote the number of days needed to collect k distinct coupons, given that you have already collected $k-1$ distinct coupons. After collecting $k-1$ distinct tokens, the probability you will collect another distinct token is $\frac{n-k+1}{n}$. Thus $X_k \sim \mathrm{Geo}(\frac{n-k+1}{n})$, so $\mathrm{E}[X_k] = \frac{n}{n-k+1}$. By the linearity of expectations,

$$\mathrm{E}[X] = \sum_{k=1}^{n} \mathrm{E}[X_k] = \sum_{k=1}^{n} \frac{n}{n-k+1} = nH_n \sim \mathcal{O}(n\ln(n)).$$

(b) Once again, let X_k be the number of days after the $(k-1)^{\text{th}}$ distinct coupon to collect the k^{th} distinct coupon. Its variance is:

$$\mathrm{Var}(X_k) = \frac{1-p}{p^2} = \frac{1-\frac{n-k+1}{n}}{\left(\frac{n-k+1}{n}\right)^2} = \frac{n(k-1)}{(n-k+1)^2}$$

[4] Of course we already discussed this in class, but good to practice!

Given that each X_k is independent,

$$\begin{aligned}
\operatorname{Var}(X) &= \sum_{k=1}^{n} \operatorname{Var}(X_k) \\
&= \sum_{k=1}^{n} \frac{n(k-1)}{(n-k+1)^2} \\
&= n \sum_{k=1}^{n} \frac{n-k}{k^2} \\
&= n^2 \sum_{k=1}^{n} \frac{1}{k^2} - n \sum_{k=1}^{n} \frac{1}{k} \\
&= n^2 \sum_{k=1}^{n} \frac{1}{k^2} - nH_n.
\end{aligned}$$

We simplify this using the Basel Problem to obtain

$$\operatorname{Var}(X) = n^2 \left(\frac{\pi^2}{6} \right) - nH_n \sim \mathcal{O}(n^2)$$

(c) Using Markov's inequality and part (a),

$$\Pr\left(X \geq 2n\ln(n)\right) \leq \frac{\mathrm{E}\left[X\right]}{2n\ln(n)} \sim \frac{n\ln(n)}{2n\ln(n)} = \frac{1}{2}.$$

(d) Using Chebyshev's inequality and part (b),

$$\begin{aligned}
\Pr\left(X \geq 2n\ln(n)\right) &= \Pr\left(X - \mathrm{E}\left[X\right] \geq 2n\ln(n) - \mathrm{E}\left[X\right]\right) \\
&\leq \Pr\left(\left|X - \mathrm{E}\left[X\right]\right| \geq 2n\ln(n) - \mathrm{E}\left[X\right]\right) \\
&= \Pr\left(\left|X - \mathrm{E}\left[X\right]\right| \geq n\ln(n)\right) \\
&\leq \frac{\operatorname{Var}(X)}{(n\ln(n))^2} \sim \frac{n^2}{n^2\ln^2(n)} \sim \mathcal{O}\left(\frac{1}{\ln^2(n)}\right).
\end{aligned}$$

Exercise 12

(Erickson) Recall that a family $\mathcal{H}$ of hash functions is *universal* if, for all distinct items $x \neq y$,

$$\Pr_{h \in \mathcal{H}}\left[h(x) = h(y)\right] \leq \frac{1}{m}$$

where m is the size of the hash table. For any fixed hash function h, a collision is an unordered pair of distinct items x and y such that $h(x) = h(y)$.

Suppose we hash a set of n items into a table of size $m = 2n$, using a hash function h chosen uniformly at random from some universal family. Assume $\sqrt{n}$ is an integer.

(a) Prove that the expected number of collisions is at most $n/4$.

(b) Prove that the probability that there are at least $n/2$ collisions is at most $1/2$.

(c) Prove that the probability that any subset of more than $\sqrt{n}$ items all hash to the same address is at most $1/2$. [*Hint:* Use part (b).]

(d) Now suppose we choose h at random from a *4-uniform* family of hash functions, which means for all distinct items w, x, y, z and all addresses i, j, k, l, we have

$$\Pr_{h \in \mathcal{H}} [h(w) = i \wedge h(x) = j \wedge h(y) = k \wedge h(z) = \ell] = \frac{1}{m^4}.$$

Prove that the probability that any subset of more than $\sqrt{n}$ items all hash to the same address is at most $\mathcal{O}(1/n)$.

[*Hint:* All four statements have short elementary proofs via tail inequalities.]

Solution

(a) Let's calculate the expected number of collisions when hashing n items. There are $\binom{n}{2}$ ways to choose a pair of items from n items. The expected number of collisions is the sum of the probabilities of collisions over all pairs, which can be written as:

$$\text{Expected number of collisions} = \binom{n}{2} \times \frac{1}{2n}$$

We know that $\binom{n}{2} = \frac{n(n-1)}{2}$. Substituting this in, we get:

$$\frac{n(n-1)}{2} \times \frac{1}{2n} = \frac{n-1}{4}$$

Since $n - 1 \leq n$, the expected number of collisions is at most $n/4$.

Note: I don't know why you would need to use a tail inequality for this part.

(b) Markov's inequality states that for any non-negative random variable X and any $a > 0$,

$$\Pr(X \geq a) \leq \frac{\mathrm{E}[X]}{a}$$

Let X be the random variable representing the number of collisions. From part (a), we know that $\mathrm{E}[X] \leq n/4$. Setting $a = n/2$, we get:

$$\Pr(X \geq n/2) \leq \frac{n/4}{n/2} = \frac{1}{2}$$

(c) Suppose some subset of at least $\sqrt{n}+1$ items all hash to the same address. Then the number of collisions is at least

$$\binom{\sqrt{n+1}}{2} = \frac{\sqrt{n+1}\sqrt{n}}{2} \geq \frac{n}{2}.$$

From part (b), the probability that there are at least $n/2$ collisions is at most $1/2$. Thus, the probability that any subset of more than $\sqrt{n}$ items all hash to the same address is at most $1/2$.

(d)

Exercise 13

Let X be the number of fixed points of a random permutation on n elements. Calculate $\mathrm{E}[X]$ and $\mathrm{Var}(X)$ and use these to bound the probability that the number of fixed points of a random permutation is at least 10. Are Chernoff bounds useful to improve this?

Solution

First, we write X as the sum of indicators, $X = \sum_{i=1}^{n} I_i$, where I_i is 1 when element i in the permutation is a fixed point. In a random permutation, i

is equally likely to be in any location, so the probability that it is in its own position $\Pr(I_i = 1) = \mathrm{E}[I_i] = 1/n$. Thus by linearity of expectation, $\mathrm{E}[X] = n\,\mathrm{E}[I_1] = 1$.

Now, these I_i are not independent random variables, so we cannot add the variances. To start, we consider $\mathrm{E}[I_i I_j]$ for $i \neq j$. $I_i I_j$ is 1 if both i and j are fixed points. There are $n(n-1)$ possible places for this pair to go, all equally likely, thus $\mathrm{E}[I_i I_j] = \frac{1}{n(n-1)}$. Thus:

$$\begin{aligned}
\mathrm{E}\left[X^2\right] &= \mathrm{E}\left[(\sum_i I_i)^2\right] \\
&= \mathrm{E}\left[\sum_i I_i^2 + \sum_{i \neq j} I_i I_j\right] && \text{(By expanding the square)} \\
&= \sum_i \mathrm{E}\left[I_i^2\right] + \sum_{i \neq j} \mathrm{E}\left[I_i I_j\right] && \text{(By Linearity of Expectation)} \\
&= \sum_i \frac{1}{n} + \sum_{i \neq j} \frac{1}{n(n-1)} \\
&= \frac{n}{n} + \frac{n(n-1)}{n(n-1)} = 2
\end{aligned}$$

Thus $\mathrm{Var}(X) = \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2 = 2 - 1^2 = 1$.

- Markov's inequality gets us that $\Pr(X \geq 10) \leq 1/10$.
- Chebychev's gets us that $\Pr(X \geq 10) \leq \Pr(|X - 1| > 9) \leq 1/81$.
- Chernoff bounds cannot be used, since the I_i are not independent.

Exercise 14

Consider the following problem P: Given a directed graph $G = (V, E)$ with weights $w_{i,j} \geq 0$ for each edge $(i, j) \in E$, the task is to partition V into two sets U and $W = V \setminus U$ so as to maximize the total weight of edges going from U to W, i.e., maximize

$$\sum_{\substack{(i,j) \in E \\ i \in U, j \in W}} w_{i,j}.$$

(a) Give a randomized $\frac{1}{4}$-approximation for the problem.

(b) Consider the following mixed integer linear program:

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} w_{i,j} z_{i,j} & \\ \text{s.t.} \quad & z_{i,j} \le x_i, & \text{for } (i,j) \in E, \\ & z_{i,j} \le 1 - x_j, & \text{for } (i,j) \in E, \\ & x_i \in \{0,1\}, & \text{for } i \in V, \\ & z_{i,j} \in [0,1], & \text{for } (i,j) \in E, \end{aligned}$$

Show that it solves P.

(c) *(Optional)* Consider a randomized rounding algorithm that solves the above mixed integer linear program and independently puts each vertex i into U with probability $\frac{1}{4} + x_i/2$. Prove that this gives a randomized $\frac{1}{2}$-approximation algorithm for P.

Solution

(a) A randomized algorithm is as follows: for each vertex $v \in V$, randomly assign it to either set U and W with equal probability.

For each edge (i,j), the probability that i is in U and j is in W is $\frac{1}{4}$. Thus, the expected weight contributed by edge (i,j) is $\frac{1}{4}w_{ij}$. Summing over all edges, the total expected weight is

$$\frac{1}{4}\sum_{(i,j)\in E} w_{ij} \ge \frac{1}{4} \sum_{\substack{(i,j)\in E \\ i\in U^\star, j\in W^\star}} w_{i,j},$$

where $U^\star, W^\star$ is an optimal partition. Thus this algorithm is an $\frac{1}{4}$-approximation.

(b) Here x_i is a binary value representing whether or not vertex i is in U. z_{ij} is an activation value (forced to be binary) representing whether or not vertex i is in U and vertex j is in V. If z_{ij} is activated, then weight w_{ij} is allowed to contribute to the final sum. Thus, maximizing this sum maximizes the total weight of edges going from U to W.

Exercise 15

(*Erickson*)

(a) The **chromatic number** $\chi(G)$ of an undirected graph G is the minimum number of colors required to color the vertices, so that every edge has endpoints with different colors. Computing the chromatic number exactly is NP-hard, because 3Color is NP-hard.

Prove that the following problem is also NP-hard: Given an arbitrary undirected graph G, return any integer between $\chi(G)$ and $\chi(G) + 473$.

(b) Let $\alpha(G)$ denote the number of vertices in the largest independent set in a graph G. Prove that the following problem is NP-hard: Given a graph G, return any integer between $\alpha(G) - 31337$ and $\alpha(G) + 31337$.

[Note: This does not contradict the possibility of constant **factor** approximation algorithms.]

Solution

(a) First note that computing $\chi(G)$ is NP-hard by a reduction from 3Color, since

$$\exists \text{ 3-coloring of } G \to \chi(G) \leq 3$$
$$\nexists \text{ 3-coloring of } G \to \chi(G) \geq 4$$

Now let G' be the union of 473 copies of G, with additional edges between *every* vertex of each copy and *every* vertex in *every* other copy. Given G, we can easily build G' in polynomial time by brute force. Let $\chi(G)$ and $\chi(G')$ denote the minimum number of colors in any proper coloring of G, and define $\chi(G')$ similarly.

$\Longrightarrow$ Fix any coloring of G with $\chi(G)$ colors. We can obtain a proper coloring of G' with $473 \cdot \chi(G)$ colors, by using a distinct set of $\chi(G)$ colors in each copy of G. Thus, $\chi(G') \leq 473 \cdot \chi(G)$.

$\Longleftarrow$ Now fix any coloring of G' with $\chi(G')$ colors. Each copy of G in G' must use its own distinct set of colors, so at least one copy of G uses at most $\left\lceil \frac{\chi(G')}{473} \right\rceil$ colors. Thus, $\chi(G) \leq \left\lceil \frac{\chi(G')}{473} \right\rceil$.

These two observations immediately imply that $\chi(G') = 473 \cdot \chi(G)$. Thus

$$\exists \text{ 3-coloring of } G' \to \chi(G') \leq 3 \cdot 473$$
$$\nexists \text{ 3-coloring of } G' \to \chi(G') \geq 4 \cdot 473,$$

so given an algorithm returning any integer between $\chi(G)$ and $\chi(G)+473$, we could compute 3Color in polynomial time.

Note: You can also do this from a reduction from computing $\chi(G)$:

"If we could compute some integer k such that $k \leq \chi(G') \leq k + 473$, then $\chi(G) = \chi(G')/473 = \lceil k/473 \rceil$. Thus, if we could compute such an integer k in polynomial time, we could compute $\chi(G)$ in polynomial time. But computing $\chi(G)$ is NP-hard!"

(b) Let G' be the union of 31337 copies of G, with additional edges added as follows: for every edge uv in G, add an edge from *every* copy of u to *every* copy of v.

$\implies$ For any independent set S in G, we can select the copies of every vertex $v \in S$ in G' to construct a independent set 31337 times larger. Thus $\alpha(G') \geq 31337 \cdot \alpha(G)$.

$\impliedby$ The largest independent set in G' can at most have a representative from each vertex of the largest independent set in G from each of the 31337 copies. Hence, $\alpha(G') \leq 31337 \cdot \alpha(G)$.

These two observations imply that $\alpha(G') = 31337 \cdot \alpha(G)$. If we could compute some integer k such that $k - 31337 \leq \alpha(G') \leq k + 31337$, then $\alpha(G) = \alpha(G')/31337 = \lceil k/31337 \rceil$. Thus, if we could compute such an integer k in polynomial time, we could compute $\alpha(G)$ in polynomial time. But computing $\alpha(G)$ is NP-hard!

Extra Practice

These are a collection of problems online I found interesting. I haven't drafted up solutions for these, and their relevance to the actual exam is undetermined.

Exercise 16

(Mount) This problem demonstrates an important application of computational geometry to the field of amphibian navigation. A frog who is afraid of the water wants to cross a stream that is defined by two horizontal lines $y = y^-$ and $y = y^+$. On the surface there are n circular lily pads whose centers are at the points $P = \{p_1, \ldots, p_n\}$. The frog crosses by hopping across the circular lily pads. The lily pads grow at the same rate, so that at time t they all have radius t (see Fig. 2(a)). Help the frog out by presenting an algorithm that in $O(n \log n)$ time determines the earliest time t^* such that there exists a path from one side of the stream to the other by traveling along the lily pads (see Fig. 1(b)).

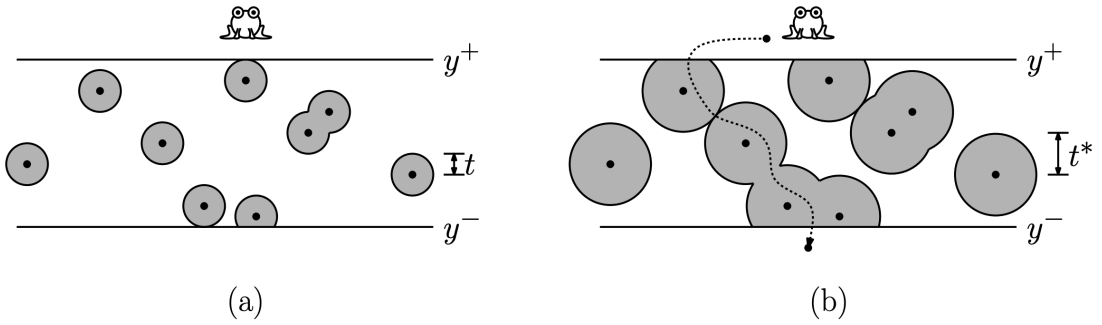


Figure 2

Exercise 17

You are given a collection of n non-intersecting circular disks in the plane, each of unit radius. Let $P = \{p_1, ..., p_n\}$ denote their center points. Given an arbitrary disk p_i in P as input, your task is to determine whether it is possible for this disk to escape from the others, meaning that it is possible to move this disk arbitrarily far away from the others without intersecting or moving any

of the other disks. Present an $O(n \log n)$ time algorithm to determine if it is possible for the disk to escape, and prove your algorithm is correct.

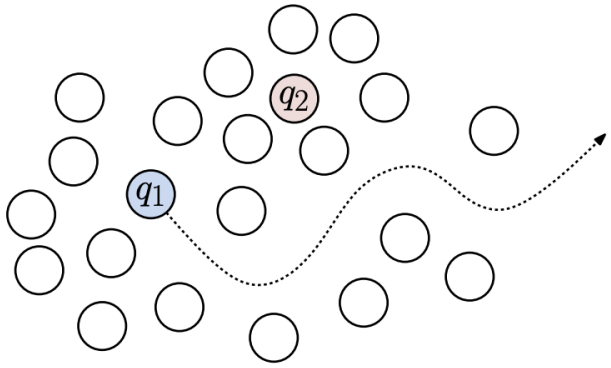


Figure 3: Here q_1 can escape, but q_2 cannot

Exercise 18

(Mount) The Delaunay triangulation of a convex polygon is defined to be the Delaunay triangulation whose sites are the vertices of the polygon. As usual, let us assume that the vertices $V = \{v_1, \ldots, v_n\}$ of the polygon are presented in counterclockwise order around the polygon. Also assume that $n \geq 3$ and no three vertices are collinear. In this problem, we will analyze a randomized incremental algorithm for constructing the Delaunay triangulation of a convex polygon.

The algorithm is similar to the randomized incremental algorithm given in class, but with the following differences. First, we do not use any sentinel sites, just the points themselves. We begin by permuting the points randomly. Let $P = (p_1, \ldots, p_n)$ denote the sequence of vertices after this permutation has been applied. We start with the triangle $\Delta p_1 p_2 p_3$. Then, we go through the points p_4 through p_n, adding each one and updating the Delaunay triangulation as we go. The insertion process for p_i involves the following steps:

i. Determine the edge ab of the current convex hull that is visible to p_i (see Fig. 1(a)).

ii. Connect p to the convex hull by adding the edges ap_i and $p_i b$ to the convex hull (see Fig. 1(b)).

iii. As in the notes, perform repeated edge flips until all the triangles incident to p_i satisfy the local Delaunay condition (see Fig. 1(c)).

Answer the following questions about this algorithm:

(a) Prove that in any triangulation of an n-sided convex polygon, the number of triangles is $n-2$ and the number of edges (including the edges of the convex hull) $2n-3$.

(b) What is the average degree of a vertex in the Delaunay triangulation of n? (Include the two edges of the convex hull that are incident to this vertex.) Derive your answer.

(c) Apply a backwards analysis to bound the expected number of edge flips performed when the ith site is inserted into the diagram (for $4 \leq i \leq n$).

(d) In step (i) of the above algorithm, we need to determine the edge ab of the convex hull that is visible to the new site p_i. Describe a method for answering these queries and derive its expected running time. (Hint: As we did in the standard Delaunay Triangulation algorithm, apply some form of bucketing combined with a backwards analysis.) $O(n \log n)$ total expected time is possible.

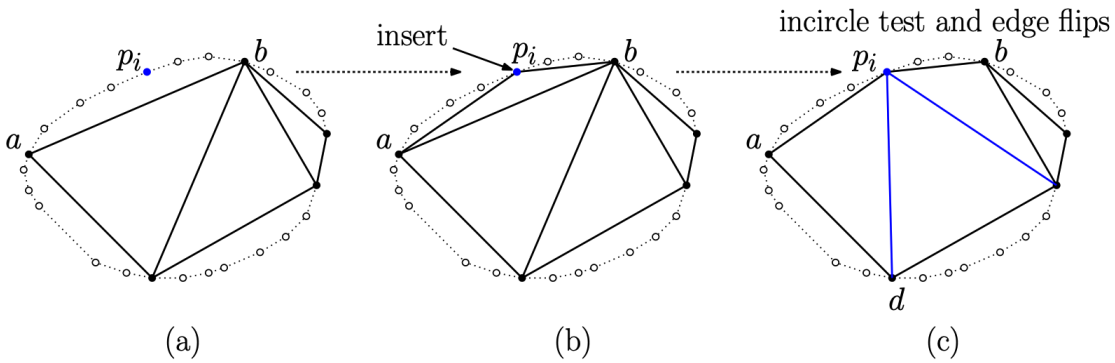


Figure 4: Delaunay triangulation of a convex point set.

Exercise 19[5]

You're the new owner of the restaurant chain Los Pollos Hermanos, and you want to expand your empire. You want to select a site for a new franchise which

[5]Yes, we covered this problem in class. But good to make sure you understand the solution.

is maximally far from all existing franchises to minimize competition and situate yourself near people who previously did not have a franchise nearby.

We can represent this problem as follows: given an n element point set P in $\mathbb{R}^2$ representing our franchises, we want to compute the largest circular disk that has its center in (or on) P's convex hull and contains no points of P in its interior (call such a disk *empty*). Describe and prove a $\mathcal{O}(n \log n)$ time algorithm that finds the empty disk of largest radius.

Hint: What can you say about the center of the disk?

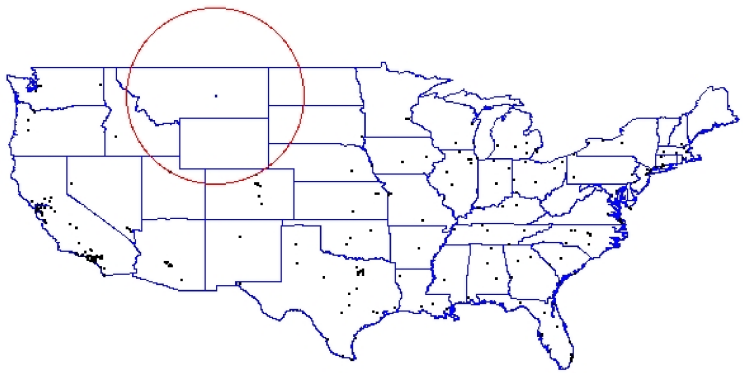


Figure 5: After your investigation, you find that your next Los Pollos Hermanos franchisee will be in Montana

Exercise 20

The following problem arises in telecommunications networks, and is known as the SONET ring loading problem. The network consists of a cycle on n nodes, numbered 0 through $n-1$ clockwise around the cycle. Some set C of calls is given; each call is a pair (i, j) originating at node i and ending at node j. The call can be routed either clockwise or counterclockwise around the ring. The objective is to route the calls so as to minimize the total load on the network. The load L_i on link $(i, i+1 \pmod n)$ is the number calls routed through the link, and the total load is $\max_{1 \le i \le n} L_i$. Give a 2-approximation algorithm for the SONET ring loading problem.