

CS157 Final

Due: Tuesday, May 5, 2009, at 2pm.

Instructions

No collaboration is allowed on exams. The only references you can use are your own notes taken in class, the class textbook, and the course website. Using any other source including online sources (e.g. Wikipedia) is not permitted.

Type up your solutions. You may leave space in your file using a command such as `vspace` and insert hand-drawn figures, if that is more convenient.

Problem 1. Set intersection

Here is an algorithm to compute the intersection of two sorted sets A and B .

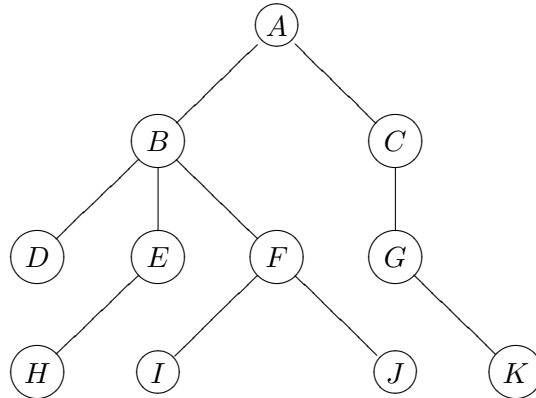
- Up to exchanging A and B , assume $|A| \leq |B|$.
- (Base case $A = \emptyset$: return the empty set.)
- Let α be the median element of A . Use binary search to search for α in B . If $\alpha \in B$, add α to the intersection I .
- Let B_L denote the elements of B less than α , B_R denote the elements of B greater than α , A_L denote the elements of A less than α , and A_R denote the elements of A greater than α .
- Recursively compute the intersection of A_L and B_L , and add the result to I .
- Recursively compute the intersection of A_R and B_R , and add the result to I .
- Output I .

- (a) Assume that the largest element of A is larger than the smallest element of B . What then is the running time of the algorithm? Prove it briefly.
- (b) Assume that $|B| = 2^k$ and $|A| = |B|/2^\ell$ consists of every 2^ℓ th element of B . What then is the running time of the algorithm? Give a proof sketch.
- (c) Which of the two above cases has a better running time? How do those runtimes compare to the merge of mergesort? Explain briefly.
- (d) (Open-ended.) Design an efficient algorithm that takes as input k sets A_1, A_2, \dots, A_k , each of size k , and decides whether their intersection $A_1 \cap A_2 \cap \dots \cap A_k$ is empty. What is your running time? You may use randomization, if that helps get a faster running time.

Problem 2: Vertex Cover

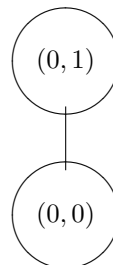
A vertex cover \mathcal{C} of a graph is a set of vertices such that each edge is incident to at least one vertex of \mathcal{C} . It is minimum if it has the smallest possible number of vertices.

- (a) Recall that a tree is a connected acyclic graph. For the following tree, find all minimum vertex covers. Please write the vertices of a vertex cover in alphabetical order, and list the vertex covers in lexicographical order.

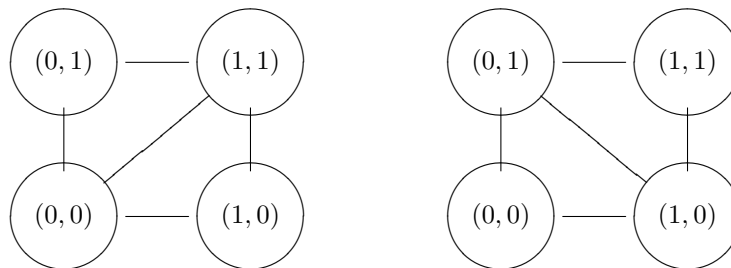


- (b) Give a linear time algorithm that takes a tree as input and returns as output the cardinality of the tree's minimum vertex cover. (Dynamic programming may be helpful, or maybe divide-and-conquer...)
- (c) Prove correctness
- (d) Briefly prove that the running time is linear.

Consider a special type of graph called a *strip*. A strip is a graph whose vertices are on a rectangular grid. A strip of depth 0 has no vertices or edges. A strip of depth 1 looks like this:

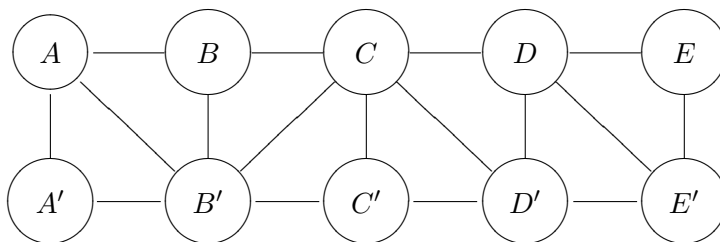


A strip of depth 2 looks like one of these two graphs:



A strip of depth n consists of vertices at position $(i, 0)$ and $(i, 1)$ for all i s.t. $0 \leq i < n$, has n vertical edges $\{(i, 0), (i, 1)\}$, $2(n - 1)$ horizontal edges $\{(i, 0), (i + 1, 0)\}$ and $\{(i, 1), (i + 1, 1)\}$, and $n - 1$ edges, where for each $i = 1, 2, \dots, n - 1$, we have either edge $\{(i - 1, 0), (i, 1)\}$ or edge $\{(i - 1, 1), (i, 0)\}$.

- (e) For the following strip, find all minimum vertex covers. Please format the vertex covers as lexicographically ordered, where $A < A' < B < B' < \dots$.



- (f) Design a linear time algorithm that takes a strip as input and returns as output the cardinality of the strip's minimum vertex cover. (No proofs required.)
- (g) (Open-ended.) What feature common to trees and strips enabled you to design a linear-time solution? Propose a conjecture generalizing both setting, of the form: "If the input graph has the following structural property: \dots , then there is a linear-time algorithm for the problem."

Problem 3: Testing wireless connections

A campus has w wireless access points and there are s students with laptops. Laptop ℓ is within the range of a set S_ℓ of access points. We assume that each laptop is within range of at least one access point, and that each access point has at least one laptop within its range. The testing problem, given w, s and the collection of sets $\mathcal{S} = (S_\ell)_{1 \leq \ell \leq s}$ satisfying the assumption, is to find a minimum set $T = T(\mathcal{S})$ of pairs (ℓ, p) , for a laptop $\ell \leq s$ and an access point $p \leq w$ within its range, such that each laptop appears in at least one pair and such that each access point also appears in at least one pair. That is, by trying out all the connections in T , we can be sure that each laptop and each access point have correctly functioning software.

- (a) Let $w = s = 5$. Give a collection \mathcal{S} such that $T = T(\mathcal{S})$ has minimum cardinality. Give a collection \mathcal{S} such that $T = T(\mathcal{S})$ has maximum cardinality. (No proofs are required.)
- (b) A matching is a set $M = M(\mathcal{S})$ of pairs (ℓ, p) , for a laptop $\ell \leq s$ and an access point $p \leq w$ within its range, such that each laptop appears in at most one pair and such that each access point also appears in at most one pair. Let $w = s = 5$. For each $x \in \{1, 2, 3, 4, 5\}$, give a collection \mathcal{S} such that $T = T(\mathcal{S})$ has minimum cardinality, among the collections \mathcal{S} whose maximum matching has size x . For each $x \leq 5$, give a collection \mathcal{S} such that $T = T(\mathcal{S})$ has maximum cardinality, among the collections \mathcal{S} whose maximum matching has size x . What do you observe? (No proofs are required.)

- (c) Conjecture an algebraic relation between the size of T , solution to the test problem, and the maximum matching size. (No proofs are required.) (You may wish to try out other values of w, s , in particular when $s \neq w$, but I do not need to see your exploratory examples.)
- (d) Prove the above algebraic relation. (Hint: induction is a possibility!)
- (e) Propose a polynomial-time algorithm to solve the test problem. (No proofs are required.)
- (f) What is its running time? Justify briefly.
- (g) Write an integer programming formulation for the test problem. Explain. Write its linear programming relaxation.
- (h) Do you think that your linear program always has an optimal solution whose variables are integers? Explain briefly (no proof required.)
- (i) We now consider a set of n devices, along with a collection \mathcal{E} of compatible pairs $\{x, y\}$, where x and y are devices. The device test problem asks for a minimum set T of compatible pairs such that every device occurs in at least one pair. Propose a simple polynomial-time approximation algorithm for this problem, and prove that the resulting T has size at most twice the optimal size.