Written Qualifying Exam
Theory of Computing

Fall 2002

Friday, September 27, 2002

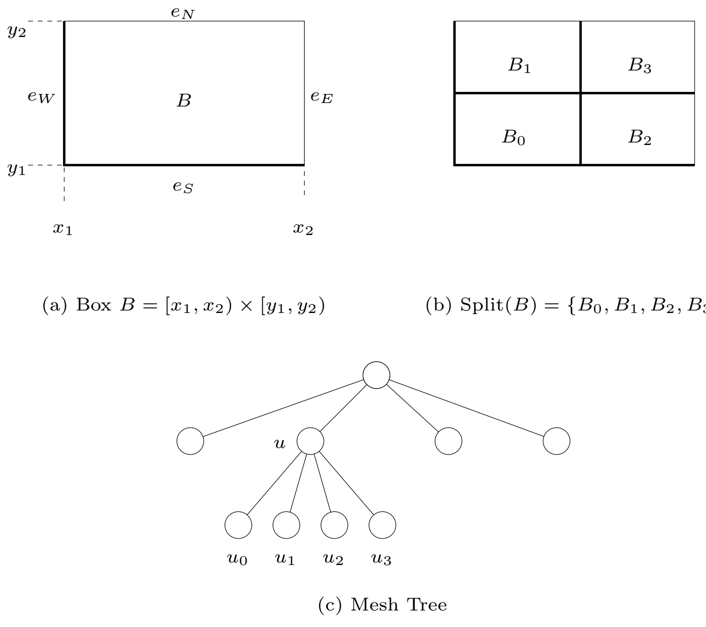
This is a *three hour* examination. All questions carry the same weight. Answer all of the following *six* questions.

• Please check to see that your name and address are correct as printed on your blue-card.
• Please *print* your name on each exam booklet. Answer each question in a *separate* booklet, and *number* each booklet according to the question.

Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results, except where asked to prove them.

PROBLEM 1 New booklet please. [10 points]

A **box** is a subset of the Euclidean plane of the form $B = [x_1, x_2) \times [y_1, y_2) \subseteq \mathbb{R}^2$ as shown Figure 1(a). It has four edges, viewed as closed line segments, named the North, South, East and West. These edges are e_N, e_S, e_E, e_W in Figure 1(a). The **split** of B is a set of 4 boxes B_0, B_1, B_2, B_3, all with the same dimensions, and such that B is the union of the B_i's.



(a) Box $B = [x_1, x_2) \times [y_1, y_2)$ (b) Split$(B) = \{B_0, B_1, B_2, B_3\}$

(c) Mesh Tree

Figure 1: Mesh Tree

A **mesh tree** T is a rooted tree in which each non-leaf u has exactly four children $u.0, u.1, u.2, u.3$ such that (a) u is associated with a box $u.Box$, and (b) if u is not a leaf, then the boxes of its children forms the split of $u.Box$.
If the index i of $u.i$ is equal to $ab \in \{0,1\}^2$ in binary, then we also write $u.i = u.ab$. Thus $u.0 = u.00, u.1 = u.01, u.2 = u.10, u.3 = u.11$. The **type** of $u.ab$ is (a, b). Also, the **x-type** of $u.ab$ is a, and the **y-type** is b. Each node u also maintains a pointer $u.p$ to its parent. When u is a leaf, we call $u.Box$ a **leaf box**. If B, B' are disjoint boxes, we say B **abuts** B' if an edge e of B is contained in an edge e' of B'. This relationship is asymmetric in general. If e is a North edge, then e' must be a South edge, and we say B abuts B' **to the north**. Similarly for the other compass directions. Assume every leaf u maintains 4 pointers $u.N, u.S, u.E, u.W$ to other leaf nodes whose boxes $u.Box$ abuts. In summary, u maintains the pointers $u.p, u.i (i = 0, \ldots, 3), u.D (D = N, S, E, W)$ where any of these pointers may be null. HINT: in the following, it may be useful to first consider the 1-dimensional version of mesh trees, as these are just binary trees.

(a) Suppose $depth(u) \geq depth(v)$ where $depth(u)$ is the length of the path from the root to u. Use node types to characterize the condition "$u.Box$ abuts $v.Box$ to the East".

(b) Give an algorithm $Split(u)$ which, given a leaf u, refines the mesh tree by creating four children for u. In particular, describe how to initialize the pointers $u.i.N, u.i.S, u.i.E, u.i.W$ at any child $u.i$ of u (you may just do it for $i = 0$). State any assumptions needed to implement your algorithm.

(c) Call a node u **smooth** if for any compass direction $D \in \{N, S, E, W\}$ such that $u.D$ is non-null, we have $depth(u) - depth(u.D) \in \{0, 1\}$. The mesh tree is **smooth** if every node in T is smooth. Give an efficient algorithm $Smooth(u)$ which, given the root u of the mesh tree, will refine the mesh tree into a smooth tree by repeated calls to $Split(w)$. Upper bound the complexity of your algorithm as a function of n, the number of leaves in the original tree.

SOLUTION

Solution (a): $u.Box$ abuts $v.Box$ to the East iff there is are two sequence of nodes $u_1, u_2, \ldots, u_k$ and $v_1, v_2, \ldots, v_\ell$ such that
(i) $u = u_1$ and $v = v_1$,
(ii) $u_i.p = u_{i+1}$ $(i = 1, \ldots, k-1)$, and $v_j.p = v_{j+1}$ $(j = 1, \ldots, \ell - 1)$,
(iii) $u_k = v_\ell$ and $depth(u) - depth(v) = k - \ell \geq 0$,
(iv) the x-type of $u_1, \ldots, u_{k-2}$ are all 1 but the x-type of u_{k-1} is 0,
(v) the x-type of $v_1, \ldots, u_{\ell-2}$ are all 0 but the x-type of $u_{\ell-1}$ is 1,
(vi) the y-type of $v_1, \ldots, v_\ell$ is equal to the y-type of $u_{k-\ell}, \ldots, u_k$.

Solution (b): To split u, we create four children $u.0, u.1, u.2, u.3$ and initialize the parent and child pointers. Consider how to initialize the pointers $u.0.N, u.0.S, u.0.E, u.0.W$. Clearly, $u.0.N = u.1$ and $u.0.E = u.2$. What about $u.0.S$ and $u.0.W$? It is simply equal to $u.S$ and $u.W$, respectively. If $u.S$ is undefined, then $u.0.S$ would be undefined.

Solution (c): Starting from node u, we use a BFS method to find all the leaves, and put them into a queue Q. This queue is just a list $Q = (L_1, L_2, \ldots, L_r)$ of lists L_i. Each list L_i contains leaves at a fixed depth d_i, and $d_1 > d_2 > \cdots > d_r$. We extract the top element of the queue, namely the first item u in list L_1, and process it as follows: check each node v that u abuts. If $depth(u) - depth(v) > 1$, and v is a leaf, then proceed to split v and move to the child c_1 of v that u abuts. We keep doing this, giving us a sequence of nodes: $c_1, c_2, \ldots, c_m$ such that $depth(u) - depth(c_m) = 1$. If $u.D = v$, we now change it to $u.D = c_m$. What if v is not a leaf? Since v was a leaf when u was originally put in the queue, this means that v has been split. So we trace down the descendents of v to find the current leaf box which u abuts. It is important to see that we do not descend below the depth of u in this process.

Note some newly created nodes may not be smooth, and so they need to be put into the queue as well. All the cascaded costs are charged to u. It is not hard to that the cost charged to any node at depth D is $O(D)$. Hence the overall time is $O(nD)$ where D is the depth of the mesh tree and there are n

leaves. But we can actually argue $O(n)$ time, by an amortized analysis.

PROBLEM 2 New booklet please. [10 points]

The Sawzall lumber yard computes the price of a wood board as follows. A board of length n is viewed as an array A of length n. Location $A[j]$ contains a number that indicates the quality of the board over the interval $(j,\ j+1)$.

The quality of the board is defined to be $\min_{0\le j<n} A[j]$. (Here larger values mean higher quality, so that the wood is rated according to its worst interval.) The price of a board of length m is equal to

$$n^2 P[\min_{0\le j<m} A[j]],$$

where P is a pricing table for the different quality ratings. The lumber yard can cut boards to produce higher quality pieces which might increase the sum of the values of the pieces even though the pieces are shorter.

However, each cut costs the company \$1.00 in expenses.

(a) Present, at as high a level as you like, a program, recursive program or system of recursive equations that can be efficiently computed to maximize the company's profit for a board. That is, you are to maximize the sum of the prices for the different pieces minus the number of cuts.

(b) State, with a brief justification, the running time of your program description.

SOLUTION

Solution (a)

Define

$$Cost(i,j) = \begin{cases} P[A[i]], & \text{if } i=j; \\ \max\{(j-i+1)^2 P[\min_{i\le k\le j} A[k]], & \\ \qquad -1+\max_{i\le k<j}\{Cost(i,k)+Cost(k+1,j)\}\}, & \text{if } i<j. \end{cases}$$

Use a lookup table in the dynamic programming to make the programming efficient. The maximum profit for the board is $C[0,n-1]$. Standard path recovery can record the locations of the cuts.

Solution (b)

The time is $\Theta(n^3)$ because there are $\Theta(n^2)$ table entries that, on average, require $\Theta(n)$ work per entry.

PROBLEM 3 New booklet please. [10 points]

Senator Gerry B. Mander recently proposed that most roads be painted red or blue. A few, he thought, should be left unpainted, with the color selection to be made at a later date.

The B-Mander single-source-shortest paths problem is the following: Let $G=(V,E=E_r\cup E_b\cup E_u,W[i,j])$ be a directed graph with positive edge costs $W[i,j]$ for $(i,j)\in E$. Let $V=\{1,2,3,\ldots,n\}$ be the vertices. Let vertex 1 be the source. Let E_r, E_b, E_u denote the red, the blue, and the uncolored edges of G, respectively.

The legal paths are sequences of edges that alternate in color beginning with red, and ending with blue, so that a permissible path could have a red edge followed by a blue, or a red-blue-red-blue sequence, etc. However, an uncolored edge could be used as a substitute for a colored edge, so that, for example, r, u, and u, b, u, u, r, u are also colorings that could be assigned to the consecutive edges in a legal path.

Formally, the colorings are sequences in the regular expression $((r+u)(b+u))^*$.

(a) Present an efficient algorithm to compute, for all destinations v, the length of the shortest legal path from vertex 1 to v in the B-Mander SSSP problem.

Note that an uncolored edge could be a substitute for an r edge in some shortest paths and a b edge in others.

(b) Explain how to implement path recovery.

SOLUTION
Solution(a)

The easiest solution is to construct an new graph H as follows.

Duplicate the vertex set, so that vertex i has its duplicate as vertex $i+n$, for $i = 1, 2, \ldots, n$. For each red edge $(i, j) \in E_r$, insert its representative edge from i to $n+j$. For each blue edge $(i, j) \in E_b$, insert its representative edge from $i+n$ to j. For each uncolored edge $(i, j) \in E_u$, insert the two representatives $(i, j+n)$, and $(i+n, j)$

Now run your favorite SSSP algorithm (such as Dijkstra's) from vertex 1.

The B-Mander SSSP distance from 1 to i, for $i \leq n$, is the computed SSSP cost from 1 to i.

Solution (b)

Path recovery is now standard, and needs no explanation. In terms of the original vertices, each vertex has two predecessors in the solution "tree." One predecessor of v would connect to v via a blue edge, and the other would connect via a red edge.

PROBLEM 4 New booklet please. [10 points]

Let E be a Boolean formula and $\#(E)$ denote the number of satisfying assignments to the variables occuring in E. Let SAT, $UNIQUE$ and $DOUBLE$ denote (respectively) the set of Boolean formulas E such that $\#(E) \geq 1$, $\#(E) = 1$ and $\#(E) = 2$.

(a) Show that $UNIQUE$ can be solved in polynomial time if we have an oracle for SAT.

(b) Repeat part (a) for $DOUBLE$ in place of $UNIQUE$.

SOLUTION
Let E_i^b denote the formula E where the ith variable is set to b. Similarly, write E_{ij}^{bc} for setting the ith and jth variables.

(a) Note that $\#(E) = 1$ iff the following is true

$$E \wedge \bigwedge_i (E_i^0 \not\equiv E_i^1)$$

(b) Note that $\#(E) \geq 3$ iff there is some $b, c \in \{0, 1\}$ and some $i < j$ such that the following formula is true:

$$E_{ij}^{bc} \wedge E_{ij}^{b\overline{c}} \wedge E_{ij}^{\overline{b}c}.$$

If $THREE(b, c, i, j)$ denotes this formula then $\#(E) = 2$ iff

$$E \wedge \neg UNIQ(E) \wedge \bigwedge_{b,c,i,j} \neg(THREE(b, c, i, j))$$

where $UNIQ(E)$ is the predicate of part (a).

PROBLEM 5 New booklet please. [10 points]

Let ϕ_i denote the ith partial recursive function, $K = \{i : \phi_i(i) \downarrow\}$ and $TOT = \{i : \phi_i \text{ is total}\}$. Show that the complement of the set TOT is K-r.e., i.e., there is an partial recursive function that uses K as oracle, and on any input n, halts iff $n \notin TOT$.

SOLUTION

An index n is in the complement of TOT if and only if there is an x such that $\Phi_n(x) \uparrow$. Using the parameter theorem, it is easy to see there is a recursive function $g(n, x)$ such that for all n, x,

$$W_{g(n,x)} = \begin{cases} \emptyset & \text{if } \Phi_n(x) \uparrow \\ \mathbb{N} & \text{else} \end{cases}$$

Thus $\Phi_n(x) \downarrow$ if and only if $g(n, x) \in K$. Hence to see if n is in the complement of TOT, we keep asking the K-oracle whether $g(n, x) \in K$ for $x = 0, 1, 2, \ldots$ until we succeed.

PROBLEM 6 New booklet please. [10 points]

TRUE or FALSE. Provide a counter-example or give a proof.

(a) $L = \{a^n b^{2n} a^n : n \in \mathbb{N}\}$ is context-free.

(b) The class of languages accepted by a "Queue Automata" is context-free. A Queue Automata (QA) is like a pushdown automata (PDA), except that it pushes and pops symbols from a first-in first-out queue. It can also detect if the queue is empty, and can make nondetermistic moves.

(c) The class in part (b) is context-sensitive, i.e., languages that are accepted by LBA's (a nondeterministic Turing machine operating in linear space).

SOLUTION

(a) FALSE: use pumping lemma to show non-context-freeness. Note that this language LOOKS like some context-free ones.

(b) FALSE: consider the language $\{w\#w : w \in \{0,1\}^*\}$. This language is not a CFL (e.g., let $w = 0^p1^p\#0^p1^p$ and apply the pumping lemma with p the pumping number). But a QA can accept it trivially. As another example a QA can accept $\{a^nb^nc^n : n \geq 0\}$, another non-CFL. You can let the QA perform a simultaneous push and pop if you like but in the presence of non-determinism, this adds nothing new.

(c) TRUE: a LBA can easily simulate a QA.