CSCI-GA.3520-001: Honors Analysis of Algorithms

Final Exam, Dec 18, 2023, 9:00am-1:00pm

- This is a four hour exam. There are six questions, worth 10 points each. Answer all questions and all their subparts.
- **No** access is allowed to textbooks, course notes, any other written or published materials, any online materials, and any other materials stored on devices.
- In the past years, to pass this exam, one needed to answer 3 or 4 problems well, instead of answering all the problems poorly.
- You must submit separate answers for separate questions.
- Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results and algorithms (i.e., those taught or referred to in class or homeworks).
- You must prove correctness of your algorithm and prove its time bound unless stated otherwise. The algorithm can be written in plain English (preferred) or as a pseudo-code.

Problem 1

A $\{0,1\}$-Integer Programming instance Φ consists of variables $x_1, \ldots, x_n$ that can take integer values 0 or 1 and a collection of m linear constraints (i.e. inequalities). For $1 \leqslant i \leqslant m$, the i^{th} constraint is

$$\sum_{j=1}^{n} a_{ij} x_j \geqslant c_i.$$

Here a_{ij}, c_i are also integers. The instance is said to be B-bounded if $|a_{ij}| \leqslant B$ for all i, j. Here B is thought of as a fixed constant (such as 10), but you are allowed to choose it as convenient. The instance has a solution if there is a $\{0,1\}$-assignment to the variables that satisfies all constraints. Let

$$B\text{-bounded-}\{0,1\}\text{-IP} = \{\Phi \mid \Phi \text{ is a } B\text{-bounded } \{0,1\}\text{-integer program that has a solution}\}.$$

Show that B-bounded-$\{0,1\}$-IP is NP-complete for some fixed constant B (that you are allowed to choose).

Problem 2

Given a directed graph $G(V, E)$ (no self-loops), a *directed walk* is a sequence of vertices

$$(v_1, v_2, \ldots, v_k)$$

such that $k \geqslant 1$ and $(v_i, v_{i+1}) \in E$ for every $1 \leqslant i \leqslant k-1$. Note that the k vertices on the walk need not all be distinct and $k = 1$ is a legitimate possibility. Two directed walks are considered different if the two corresponding sequences are different.

Assume that $G(V, E)$ is given in its adjacency list representation and $|V| = n$.

a. Design a $O(|V| + |E|)$ time algorithm that:

- Outputs YES if there are at least 2^n different directed walks in the graph.
- Outputs NO if the number of different directed walks in the graph is at most $2^n - 1$. In this case, the algorithm also outputs the exact number of different directed walks.

b. Give an example of a n-vertex directed graph that has exactly $2^n - 1$ different directed walks.

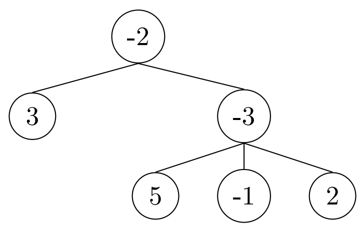
Problem 3

Let $T(V, E)$ be a tree on n vertices. Recall that a tree is a connected, undirected graph with no cycles.

A subset of vertices $Z \subseteq V$ is called *short-gapped* if for any two vertices $x, y \in Z$, on the unique path from x to y in the tree T, no two consecutive vertices are outside of Z. In other words, for any two vertices $x, y \in Z$, if $x = v_1, v_2, \ldots, v_{k-1}, v_k = y$ is the unique x to y path in T, then for every $i, 1 \leqslant i \leqslant k-1$, either $v_i \in Z$ or $v_{i+1} \in Z$ (or both).

Now suppose that every vertex $v \in T$ has an associated integer weight $\mathsf{weight}(v)$, which could be zero, positive, or negative. Give a polynomial time algorithm to find a subset Z of vertices that is short-gapped and has maximum total weight.

Example: In the tree below, the max-weight short-gapped subset consists of the vertices with weights $3, -2, 5, 2$.



Hint: Assume, without loss of generality, that the tree is rooted (as in the example above).

Problem 4

A Horn-3SAT instance is a specialized form of a 3SAT instance where each clause can have at most one negated variable and a clause can have one, two, or three literals. That is, the instance consists of Boolean variables $x_1, \ldots, x_n$ and m clauses where each clause is of one of six types (the indices i, j, k are distinct):

$$x_i, \qquad \overline{x}_i, \qquad x_i \vee x_j, \qquad \overline{x}_i \vee x_j, \qquad x_i \vee x_j \vee x_k, \qquad \overline{x}_i \vee x_j \vee x_k.$$

a. Give a polynomial time algorithm to decide whether a given Horn-3SAT instance has a satisfying assignment.

   Hint: The algorithm could proceed depending on whether there is a clause of the type $\overline{x}_i$.

b. Suppose that a Horn-3SAT instance has only three types of clauses (again, the indices i, j, k are distinct):

$$\overline{x}_i, \qquad x_i \vee x_j, \qquad x_i \vee x_j \vee x_k,$$

   and of each of these three types, there are exactly $\frac{m}{3}$ clauses. Show that there exists an assignment to the variables that satisfies at least a β fraction of clauses where $0 < \beta < 1$ is a constant that you must explicitly specify. For full credit, you need to give the largest such value of β (but no need to prove that this is indeed the largest such value).

Problem 5

a. Let A be an $n \times n$ matrix, which has at most r_a non-zero entries. Design a scheme, possibly randomized, to store the matrix and support the following operations.

- Update entry (i, j) in expected $O(1)$ time. Here *update* should allow changing the entry, adding the entry if not present already, or deleting the entry (which amounts to making it zero).
- List the non-zero entries in row i in worst case time

  $$O(\text{the number of non-zero entries in row } i).$$

  The listing need not be in row order.
- List the non-zero entries in column j in worst case time

  $$O(\text{the number of non-zero entries in column } j).$$

  The listing need not be in column order.

In addition, your data structure must use at most $O(r_a + n)$ space. In principle, the numbers r_a and r_{ai} (introduced below) could keep changing with additions and deletions, but do not worry about that.

b. Let B be a second $n \times n$ matrix, which has at most r_b non-zero entries. Suppose B is stored in the same format as matrix A. Show how to compute the matrix product $C = AB$ in expected time

$$O\big(n^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ai}\big) = O(n^2 + n \cdot r_a),$$

where r_{ai} is the number of non-zero entries in A's i^{th} row. C needs to be stored in the same format as matrices A and B.

Problem 6

Consider the following sorting algorithm. You can assume that all items are distinct.

> Input: a set of $n_s \geqslant 0$ sorted items and a set of n_u unsorted items.
>
> If $n_u = 0$ then return the sorted set. Otherwise:
>
> **Case 1**. $n_s \geqslant n_u$.
> Then let p be the middle item in the sorted set. Partition the unsorted items according to whether they are less than p **or** greater than or equal to p.
> Recurse on the following two subproblems: the first, comprising the sorted and unsorted items less than p, and the second, comprising the sorted and unsorted items greater than or equal to p.
> Return the sorted set that is ordered as the solution to the first subproblem and then the solution to the second subproblem.
>
> **Case 2**. $1 \leqslant n_s < n_u$.
> Then create a subproblem S consisting of the n_s sorted items and n_s of the unsorted items. Solve S recursively. The result is a set of $2n_s$ sorted items and $n_u - n_s$ unsorted items. Solve it recursively.
>
> **Case 3**. $n_s = 0$.
> Then take the first unsorted item, and make it into a 1-item sorted set, leaving the remaining $n_u - 1$ items as the unsorted set. Solve the resulting problem recursively.

Prove that this algorithm, to sort an initially unsorted set of size n, makes at most $O(n \log n)$ comparisons. Note that comparisons are made only in Case 1.

Hint: (a) Bound the number of comparisons that can be made by an item before it becomes part of a sorted subset. Note that once an item becomes part of a sorted subset, it remains in a sorted subset henceforth. (b) It will be helpful to measure the size of a subproblem as $n_s + n_u$. How does the size of the subproblem to which an unsorted item belongs change in the various cases?