Practice Quiz 2

Problem Practice-1. LRU data structure

Managing the primary memory of a virtual-memory system with LRU page replacement can be viewed as a dynamic data structure problem. The data structure manages a set $S = \{1, \ldots s\}$ of s fixed-size slots of primary memory. The virtual memory can be viewed as a set $P = \{1, \ldots, p\}$ of virtual-memory pages. At any time a subset of P having size at most s is resident in the s slots of primary memory. The job of the data structure is to maintain a dynamic mapping so that the system can identify whether a user's reference to a page resides in a slot of primary memory, and if not, drop the least-recently used page and replace it with the referenced page.

The LRU data structure must therefore support the following operations:

- INSERT(q, t): Insert virtual page $q \in P$ into slot $t \in S$ of primary memory. The slot t must be empty.
- USE(q): Return the slot $t \in S$ containing virtual page $q \in P$, or else NIL if q does not reside in primary memory.
- DROP$()$: Remove the least-recently used page $q \in P$ from its slot $t \in S$, and return the now-empty slot t.

Describe briefly why these operations suffice to implement the LRU page replacement policy. Give an efficient implementation of this dynamic set.

Problem Practice-2. Random-number generation

The array $A[1 \mathinner{.\,.} n]$ contains a probability distribution over the set $\{1, 2, \ldots, n\}$; that is, we have $A[i] \geq 0$ and $\sum_{i=1}^{n} A[i] = 1$. We wish to generate a random integer X in the range $1 \leq X \leq n$ such that

$$\Pr\{X = i\} = A[i] .$$

A uniform random-number generator UNIFORM() is available which generates in constant time a real number y uniformly in the range $0 \leq y < 1$. Using UNIFORM() as a subroutine, devise an efficient algorithm to generate a random integer according to the distribution specified by A.

Your algorithm may include an initialization phase to preprocess the array A. After the preprocessing phase, the user can make any number of calls to your random-number generator, each of which should return a random integer according to the distribution A. The highest priority in your design is to make your random-number generator run as fast as possible, but your preprocessing phase should be efficient as well. Analyze both the time for preprocessing and the time for actual random-number generation.

Problem Practice-3. Music recognition

A ***recorder*** is a simple blown musical instrument that sounds much like a flute. Prof. Cary Oki has recently programmed his computer to listen to a stream of music from a recorder and convert it into a set $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ of (possibly overlapping) time intervals, where each interval corresponds to the duration of a note transcribed by the system. Moreover, the professor has developed a heuristic that gives for each interval $T_i \in \mathcal{T}$, a metric m_i indicating how likely it is that note i was played by the recorder. The larger m_i, the greater the confidence that note i was played by the recorder.

(a) The professor would like to determine which notes are played by the recorder. Since the recorder can produce only one note at a time (when properly played), if two intervals have a common intersection, one of the notes must be spurious (produced by background noise). Give an efficient algorithm to determine a set $\mathcal{S} \subseteq \mathcal{T}$ of nonoverlapping intervals (ostensibly corresponding to the notes played by the recorder) that maximizes

$$\sum_{T_i \in S} m_i \ .$$

(b) The professor now wishes to extend his algorithm to recorder quartets, which consist of soprano, alto, tenor, and bass instruments. He upgrades his heuristic to give for each interval $T_i \in \mathcal{T}$, a metric m_{ik} indicating how likely it is that note i was played by the recorder k for $k = 1, 2, 3, 4$. Give an efficient algorithm to determine four disjoint sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \subseteq \mathcal{T}$, where each S_k contains nonoverlapping intervals (ostensibly corresponding to the notes played by the four respective recorders), that maximizes

$$\sum_{k=1}^{4} \sum_{T_i \in S_k} m_{ik} \ .$$

Problem Practice-4. Party, party, party!

Students at the Monotonic Institute of Technology are reluctant to go to a party if they don't know many people there. Moreover, cycles of indecision lead to situations where Alice will go to a party if Bob goes and Bob will go if Alice goes, but neither ends up going, since neither knows the other's conditions for attending.

To encourage more social behavior, the Student Invitational Party Board (SIPB) is developing a web service to help organize party going. For a given party, each student registers if he definitely wishes to attend or if he conditionally wishes to attend depending on whether a sufficient quorum of friends also attends. Specifically, the student indicates his condition on a SIPB web form by giving a threshold $t \geq 0$ and a list L of t or more other students such that he agrees to attend if at least t of the students on L also attend. Some of the students on L may not register, in which case we assume that they will not attend. At some designated time before the party, the SIPB service emails a message to each registered student whether the student should attend. The service guarantees that if all students who are emailed positive responses attend, then all the attendees' conditions are satisifed. We assume that a student is honor-bound to attend if his condition is satisfied, and that he doesn't register for conflicting parties.

The SIPB party service wishes to process the database of conditions so that as many people go to a given party as possible. Thus, in the Alice and Bob example, both should be sent a positive response. Model the problem formally, and give an efficient algorithm to select as many party-goers as possible subject to the students' conditions. For bonus points, devise a more general set of conditions that can be efficiently processed by a similar algorithm.

Problem Practice-5. Reliable distribution

A communication network consists of a set V of nodes and a set $E \subset V \times V$ of directed edges (communication links). Each edge $e \in E$ has a ***weight*** $w(e) \geq 0$ representing the cost of using e. A ***distribution*** from a given source $s \in V$ is a set of directed $|V| - 1$ paths from s to each of the other $|V| - 1$ vertices in $V - \{s\}$. The ***cost*** of a distribution is the sum of the weights of its constituent paths. (Thus, some edges may be counted more than once in the cost of the distribution.)

(a) Give an efficient algorithm to determine the cheapest distribution from a given source $s \in V$. You may assume all nodes in V are reachable from s.

(b) One of the edges in the communication network may fail, but we don't know which one. Give an efficient algorithm to determine the maximum amount by which the cost of the cheapest distribution from s might increase if an adversary removes an edge from E. (The cost is infinite if the adversary can make a vertex unreachable from s.)