Homework due Tues 11/9

- CLRS 16-2 (scheduling)
- CLRS 17-2 (binary search)

Matroids

A matroid is a pair $(\mathcal{S}, \mathcal{I})$ such that

- $\mathcal{S}$ is a finite nonempty set and
- $\mathcal{I}$ is a nonempty family of subsets of S with the following properties:

1. **heredity** If $A \in \mathcal{I}$, then every subset of A is in $\mathcal{I}$.
2. **exchange property** If $A, B \in \mathcal{I}$ and $\|\mathcal{A}\| < \|\mathcal{B}\|$, then there exists some $x \in \mathcal{B} \setminus \mathcal{A}$ such that $A \cup \{x\} \in \mathcal{I}$.

An element x is called an **extension** of $A \in \mathcal{I}$ if $A \cup \{x\} \in \mathcal{I}$.

A **maximal** element of $\mathcal{I}$ is one with no extension.

Note that, assuming heredity, the exchange property is equivalent to:

- If $A, B \in \mathcal{I}$ and $\|A\| < \|B\|$, then there is some $C \subseteq B \setminus A$ such that $A \cup C \in \mathcal{I}$ and $\|C\| = \|B\| - \|A\|$.

By the exchange property,

- **all maximal elements in a matroid have the same size.**

Graphic Matroids

For an undirected graph $G = (V, E)$, the **graphic matroid** $M_G = (\mathcal{S}_G, \mathcal{I}_G)$ of G is defined as follows:

- $\mathcal{S}_G = E$.
- $A \subseteq E$ belongs to $\mathcal{I}_G$ if and only if A is acyclic ((V, A) is a forest).

Properties of Graphic Matroids

Lemma A Let G be a connected undirected graph. Then the maximal elements of M_G are the spanning trees of G.

Proof Let $n = \|V\|$. Let T be an arbitrary spanning tree of G and let A be the edge set of T. Since T is acyclic, A belongs to $\mathcal{G}$. There cannot be elements in $\mathcal{I}$ having size n, since they induce subgraphs of G with cycles. So, A is maximal. On the other hand, if A is an element in $\mathcal{I}$ having size $n-1$, then since A induces an acyclic graph, it induces a spanning tree. ∎

Lemma B Let G be an undirected graph. Let $C_1, \cdots, C_k$ the connected components of G. For each i, $1 \leq i \leq k$, Z_i be the set of all edges of the spanning trees of C_i. Then the maximal elements of M_G is

$$Z_1 \times \cdots \times Z_k,$$

that is, the cartesian product of $Z_1, \cdots, Z_k$. ∎

Theorem C Let G be an undirected graph. Then M_G is a matroid.

Proof Let $G = (V, E)$ be an arbitrary undirected graph and let $n = \|V\|$.

Heredity Let $A \subset E$ be such that (V, A) is acyclic. Then, for all $B \subseteq A$, (V, B) is acyclic. so, M_G is hereditary.

Exchange Property Let $C_1, \ldots, C_k$ be the connected components of G. Let $A \in \mathcal{I}$. Then A can be expressed as the disjoint union of some $A_1, \ldots, A_k$ such that for each i, $1 \leq i \leq k$, A_i induces an acyclic subgraph of C_i. Let $B \in \mathcal{I}$. Then B can be similarly decomposed into the disjoint union of some $B_1, \ldots, B_k$.

Assume $\|A\| < \|B\|$. Then there is some i, $1 \leq i \leq k$, such that $\|A_i\| < \|B_i\|$. Pick such an i.

Since B_i induces an acyclic subgraph of C_i, A_i induces a forest, but not a tree, of C_i. Let $D_1, \ldots, D_m$ be the connected components of C_i that A_i induces, where $m \geq 2$. Let $t_1, \ldots, t_m$ be the number of nodes in $D_1, \ldots, D_m$. For each j, $1 \leq j \leq m$, A_i induces a tree of D_j. So, the size of A_i is $(t_1 - 1) + \cdots (t_m - 1)$.

We claim that there exist some j, j', $1 \leq j < j' \leq m$, such that B_i has an edge connecting a node in D_j and $D_{j'}$.

To prove the claim, assume otherwise. Then each edge of B_i belongs to one of the connected components $D_1, \ldots, D_m$. Since B_i induces an acyclic graph, for all j, $1 \leq j \leq m$, the number of edges of B_i within D_j is at most $t_j - 1$. So, the size of B_i is at most $(t_1 - 1) + \cdots (t_m - 1)$, and thus, does not exceed the size of A_i. This is a contradiction. This proves the claim.

By the claim, there exist some j, j', $1 \leq j < j' \leq m$, such that B_i has an edge connecting a node in D_j and $D_{j'}$. Pick such j and j' and such an edge e. Add e to A to obtain A'. By the way e is selected, A' induces an acyclic graph. So, M_G has the exchange property. ∎

Weighted Matroids

Let w be a function from $\mathcal{S}$ to $\mathcal{N}^+$, the set of all positive integers. For each $A \subseteq \mathcal{S}$, define $w(A)$ as $\sum_{x \in A} w(x)$. Call w a **weight function** of M. An **optimal** subset of M with respect to w is the one having the largest weight.

By definition, optimal subsets are maximal.

Finding an optimal subset

$\textbf{Greedy}(M, w)$
1 sort the elements in $\mathcal{S}$ in the non-increasing order of their weights;
  let $x_1, \ldots, x_m$ be the enumeration
2 $A \leftarrow \emptyset$
3 **for** $i \leftarrow 1$ **to** m **do**
4 **if** $A \cup \{x_i\} \in \mathcal{I}$
5 **then** $A \leftarrow A \cup \{x_i\}$
6 **return** A

Why does this algorithm work?

Lemma D Let k be the smallest i such that $\{x_i\} \in \mathcal{I}$. Then there is an optimal subset containing x_k.

Proof

Let B be an optimal subset with $x_k \notin B$.

No element y of B has $w(y) > w(x_k)$

Begin with $A = \{x_k\}$, add from $B - A$ until $\|A\| = \|B\|$.

$A = B - y \cup x$ for some $y \in B$

$$\begin{aligned} w(A) &= w(B) - w(y) + w(x) \qquad (1) \\ &\geq w(B) \end{aligned}$$

If an element is not an option initially, it cannot be an option later, because of heredity.

Application of the Matroid Theory: The task-scheduling problem

Objects with deadlines and penalty.

Input Integers n, $d_1, \ldots, d_n, w_1, \ldots, w_n$.

Output Find a permutation $p_1, \ldots, p_n$ of $1, \ldots, n$ that minimizes

$$\sum_{p_i > d_i} w_i.$$

Intuitively, think of $1, \ldots, n$ as n tasks to be fulfilled that require a unit-time each and of $d_1, \ldots, d_n$ as the deadlines for their tasks. Starting from time 0, the n tasks are executed in an order. If a task is not accomplished on or before its deadline, the penalty associated with it is imposed. $w_1, \ldots, w_n$ are the penalty values. The goal is to find scheduling of the tasks that minimizes the total penalty.

The matroid over the set of tasks

Let $\mathcal{S} = \{1, \ldots, n\}$

Assume that the tasks are enumerated so that their deadlines are non-decreasing.

Let $A \subseteq S$. We say that A is **good** if there is a canonical ordering of the items in A such that for all $i \in A$, the order of i in the ordering is at most d_i.

Let $\mathcal{I}$ be the set of all subsets of $\mathcal{S}$ that are good and let $M = (\mathcal{S}, \mathcal{I})$.

Theorem E M is a matroid.

Proof To prove the heredity, let $A \in \mathcal{I}$. Let π be an ordering of A such that for all $i \in A$, $\pi(i) \leq d_i$. Let B be a proper subset of A. Let σ be the ordering defined for all $i \in B$ by:

$$\sigma(i) = 1 + \|\{j \in B \mid \pi(j) < \pi(i)\}\|$$

Then, for all i, $\sigma(i) \leq \pi(i)$, and thus, $\sigma(i) \leq p_i$. So, B is good.

Exchange Property

For each t, $1 \leq t \leq n$, $A \subseteq \mathcal{S}$, let $\mu(A,t)$ be the number of $i \in A$ such that $d_i \leq t$.

To prove the exchange property of M, let A and B be good ones such that $\|A\| < \|B\|$. Since A and B are good, for all t, $1 \leq t \leq n$, both $\mu(A,t)$ and $\mu(B,t)$ are at most t. Let $t_0 = \max\{t \mid \mu(A,t) = t\}$ if there is at least one t such that $\mu(A,t) = t$ and 0 otherwise.

Claim F There is some $j \in B \setminus A$ such that $d_j \geq t_0 + 1$.

Proof Let $A_0 = \{i \in A \mid d_i \leq t_0\}$ and $A_1 = A - A_0$. Similarly, define B_0 and B_1. Since $\mu(A,t_0) = t_0$, $\|A_0\| = t_0$. Since $\|B_0\| \leq t_0$, this implies that $\|A_0\| \leq \|B_0\|$. Sicne $\|A\| < \|B\|$, this implies that $\|A_1\| < \|B_1\|$. By definition, every element of B_1 has a deadline greater than t_0, so there is some $j \in B_1 \setminus A_1$ such that $d_j \geq t_0 + 1$. ∎

The proof continues ...

Let j be such that $j \in B_1 \setminus A_1$ and $d_j \geq t_0 + 1$.

Note that, for all $i \in A_0$ $\pi(i) \leq t_0$ and for all $i \in A_1$ $d_i \geq i + 1$. This means that for each element i in A_1 can be delayed by one unit-time. Since $d_j \geq t_0 + 1$, we can add j to A and still execute all of them by their deadlines. ∎

Solution to the Maximization Problem

Let $R = w_1 + \cdots w_n$. For each good A, let $w(A) = \sum_{i \in A} w_i$ and let $Q(A) = R - w(A)$. Then the problem of minimizing the total penalty incurred is equivalent to the problem of maximizing Q, which can be solved by greedy.

Chapter 17: Amortized Analysis

Efficiency averaged over time.

We assume that a sequence of operations is executed on a data structure and calculate the cost per operation averaged over the sequence.

Some operations are cheap and some are expensive depending on the situation.

Our first example is **Multipop**,
a new operation on a stack.

With this you are able to pop
any number of elements from
a stack.

However, it is implemented by
repeated execution of **Pop**.

What are the other
permissible operations?

Creation of an empty stack,
Push, **Pop**, *and* **Empty**, *which*
tests the emptiness.

Ostensibly $\mathbf{Multipop}$ *is quite expensive because elimination of* k *objects requires* $O(k)$ *steps.*

However, for us to be able to eliminate k *objects* $\mathbf{Push}$ *has to be executed at least* k *times prior to that...*

Which means a bad thing does not happen so very often...

Our next example is a k-bit binary counter.

Suppose we will increment n times a k-bit counter that is initially set to 0...

The number of bit operations required is high if there is a long run of 1's at the lower bits of the counter, but that does not happen very often.

Amortized Analysis

Suppose that n operations chosen from $\mathbf{Pop}$, $\mathbf{Push}$, and $\mathbf{Multipop}$ are executed on an initially empty stack. The total cost for $\mathbf{Multipop}$ is the linear function of the total number of $\mathbf{Push}$, which is at most n. So, the amortized cost of $\mathbf{Multipop}$ is $O(1)$.

Suppose that a k-bit counter initially set to 0 is incremented n times. The total number of bit flips on the counter is

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n.$$

So the amortized cost is $O(1)$.

This calculation method is called **aggregate method**.

The potential method

Policy: For each i, $1 \leq i \leq n$, let c_i be the actual cost of the i-th operation and D_i be the data structure when the i-th operation has been done.

Pick a **potential function** Φ that assigns a value to the data structure and define the amortized cost $\widehat{c_i}$ as $c_i + \Phi(D_i) - \Phi(D_{i-1})$. Let $T(n) = \sum_{i=1}^{n} \widehat{c_i}$ be the total amortized cost. Then

$$\begin{aligned} T(n) &= \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) \\ &= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0). \end{aligned}$$

We'll pick Φ so that

- for all i $\Phi(D_i) \geq \Phi(D_0)$ and
- $\sum_{i=1}^{n} \widehat{c_i}$ is easy to compute.

Then $T(n)/n$ gives an upper-bound for the amortized cost. So, we will evaluate $\widehat{c_i}$ instead of c_i.

A. Stack: Define $\Phi(D_i)$ to be the stack size. Then $\Phi(D_0) = 0$, and so, for all $i \geq 1$, $\Phi(D_i) \geq \Phi(D_0)$.

The amortized cost $\widehat{c_i}$ is $1 + 1 = 2$ for $\mathbf{Push}$ and 0 for both $\mathbf{Pop}$ and $\mathbf{Multipop}$.

B. Counter: Define $\Phi(D_i)$ to be the number of bits 1 in the counter after the i-th incrementation. Then $\Phi(D_0) = 0$ and for all $i \geq 0$ it holds that $\Phi(D_i) \geq 0$.

Define t_i to be the number of bits that are reset at the i-th operation. Then for all $i \geq 0$, $\Phi(D_{i+1}) = \Phi(D_i) - t_i + 1$. Then $c_i = t_i + 1$ and $\widehat{c_i} \leq t_i + 1 + (1 - t_i) = 2$. So, the amortized cost is $O(1)$.

Dynamic Tables

A **dynamic table** is a table of variable size, where an **expansion** (or a **contraction**) is caused when the load factor has become larger (or smaller) than a fixed threshold.

Let the expansion threshold be 1 and the expansion rate be 2; i.e., **the table size is doubled when an item is to be inserted when the table is full**.

Let the contraction threshold be $\frac{1}{4}$ and the contraction rate be $\frac{1}{2}$; i.e., **the table size is halved when an item is to be eliminated when the table is exactly one-fourth full**.

Implementation of Expansion & Contraction

When these operations take place we **create a new table** and move all the elements from the old one to the new one.

Suppose that there are n calls of insertion and deletion are made, what is the average cost of each operation?

If the size is kept the same the cost is $O(1)$.

If the size is doubled from M to $2M$, the actual cost is $M + 1$. The time that it takes for the next table size change to occur is **at least M steps for doubling** and **at least $M/2$ steps for halving**. So the actual cost can be spread over the next $M/2$ "normal" steps. This gives an amortized cost of $O(1)$.

If the size is halved from M to $M/2$, the actual cost is $M/4$. The time that it takes for the next table size change to occur is **at least $\frac{M}{4}$ steps for doubling** and **at least $\frac{M}{8}$ steps for halving**. So the actual cost can be spread over the next $M/8$ steps to yield an amortized cost of $O(1)$.

Amortized Cost Analysis Using the Potential Method

For each i, $1 \leq i \leq n$, define c_i to be the number of insertions and deletions that are executed at the i-th operation, and define

$$\Phi_i \stackrel{\text{def}}{=} \begin{cases} 2num_i - size_i & \text{if } \alpha_i \geq \frac{1}{2}, \\ \frac{size_i}{2} - num_i & \text{if } \alpha_i < \frac{1}{2}, \end{cases}$$

Here $size_i$ is the table size, num_i is the number of elements in the table, and α_i is the ratio $\frac{num_i}{size_i}$ after the i-th operation. Note that

- at time 0, the table is empty, so $\Phi_0 = 0$,
- for all i, $\Phi_i \geq 0$, and thus, $\Phi_n \geq \Phi_0$, and
- $\Phi_n \leq 2n - n = n$, so the contribution of the potential function to the amortized cost is at most 1.

The Amortized Cost $\widehat{c_i}$ for Insertion Here $m = num_{i-1}$ and $s = size_{i-1}$

(a) $\alpha_{i-1} = 1$: Here $m = s$.

c_i	Φ_i	Φ_{i-1}	$\widehat{c_i}$
$m+1$	$2(m+1)-2s$	$2m-s$	3

(b) $\frac{1}{2} \leq \alpha_{i-1} < 1$:

c_i	Φ_i	Φ_{i-1}	$\widehat{c_i}$
1	$2(m+1)-s$	$2m-s$	3

(c) $\alpha_i = \frac{1}{2}$: Here $m+1 = \frac{s}{2}$.

c_i	Φ_i	Φ_{i-1}	$\widehat{c_i}$
1	$2(m+1)-s$	$s/2-m$	0

(d) $\alpha_i < \frac{1}{2}$:

c_i	Φ_i	Φ_{i-1}	$\widehat{c_i}$
1	$s/2-m-1$	$s/2-m$	0

So the amortized cost of insertion is $O(1)$.

The Amortized Cost $\widehat{c}_i$ for Deletion

(a) $\alpha_i \geq \frac{1}{2}$:

c_i	Φ_i	Φ_{i-1}	$\widehat{c}_i$
1	$2(m-1)-s$	$2m-s$	-1

(b) $\alpha_{i-1} = \frac{1}{2}$: Here $2m = s$.

c_i	Φ_i	Φ_{i-1}	$\widehat{c}_i$
1	$\frac{s}{2}-(m-1)$	$2m-s$	2

(c) $\frac{1}{4} < \alpha_{i-1} \leq \frac{1}{2}$:

c_i	Φ_i	Φ_{i-1}	$\widehat{c}_i$
1	$s/2-(m-1)$	$s/2-m$	2

(d) $\alpha_{i-1} = \frac{1}{4}$: $m = \frac{s}{4}$ and $\alpha_i < \frac{1}{2}$.

c_i	Φ_i	Φ_{i-1}	$\widehat{c}_i$
m	$s/4-(m-1)$	$s/2-m$	1

So the amortized cost of deletion is $O(1)$.