Chapter 32

Exercises - Prerequisites

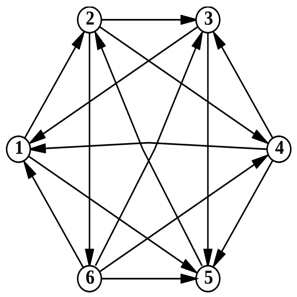
By Sariel Har-Peled, September 30, 2009[1] Version: 1.0

This chapter include problems that are perquisite. Their main purpose is to check whether you are read to take the 473 algorithms class. If you do not have the prerequisites it is *your responsibility* to fill in the missing gaps in your education.

32.1 Graph Problems

Exercise 32.1.1 (A trip through the graph.) [20 Points]

A *tournament* is a directed graph with exactly one edge between every pair of vertices. (Think of the nodes as players in a round-robin tournament, where each edge points from the winner to the loser.) A *Hamiltonian path* is a sequence of directed edges, joined end to end, that visits every vertex exactly once. Prove that every tournament contains at least one Hamiltonian path.



A six-vertex tournament containing the Hamiltonian path $6 \to 4 \to 5 \to 2 \to 3 \to 1$.

Exercise 32.1.2 (Graphs! Graphs!) [20 Points]

A *coloring* of a graph G by α colors is an assignment to each vertex of G a color which is an integer between 1 and α, such that no two vertices that are connected by an edge have the same color.

(A) **[5 Points]** *Prove* or *disprove* that if in a graph G the maximum degree is k, then the vertices of the graph can be colored using $k + 1$ colors.

(B) **[5 Points]** Provide an efficient coloring algorithm for a graph G with n vertices and m edges that uses at most $k + 1$ colors, where k is the maximum degree in G. What is the running time of you algorithm, if the graph is provided using adjacency lists. What is the running time of your algorithm if the graph is given with an adjacency matrix. (Note, that your algorithm should be as fast as possible.)

(C) **[5 Points]** A directed graph $G = (V, E)$ is a *neat* graph if there exist an ordering of the vertices of the graph $V(G) = \langle v_1, v_2, \ldots, v_n \rangle$ such that if the edge $\left(v_i \rightarrow v_j\right)$ is in $E(G)$ then $i < j$.

Prove (by induction) that a DAG (i.e., directed acyclic graph) is a neat graph.

(D) **[5 Points]** A cut (S, T) in a directed graph $G = (V, E)$ is a partition of V into two disjoint sets S and T. A cut is *mixed* if there exists $s, s' \in S$ and $t, t' \in T$ such that $(s \rightarrow t) \in E$ and $(t' \rightarrow s') \in E$. Prove that if all the non-trivial cuts (i.e., neither S nor T are empty) are mixed then the graph is not a neat graph.

Exercise 32.1.3 (Mad Cow Disease) [20 Points]

In a land far far away (i.e., Canada), a mad cow decease was spreading among cow farms. The cow farms were, naturally, organized.as a $n \times n$ grid. The epidemic started when m contaminated cows were dlivered to (some) of the farms. Once one cow in a farm has Mad Cow disease then all the cows in this farm get the disease. For a farm, if two or more of its neighboring farms have the disease than the cows in the farm would get the disease. A farm in the middle of the grid has four neighboring farms (two horizontally next to it, and two verticality next to it). We are interested in how the disease spread if we wait enough time.

- **[5 Points]** Show that if $m = n$ then there is a scenario such that all the farms in the $n \times n$ grid get contaminated.
- **[15 Points]** Prove that if $m \leq n - 1$ then (always) not all the farms are conaminated.

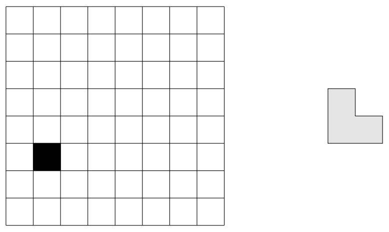
Exercise 32.1.4 (Connectivity and walking.) [10 Points]

(A) Use induction to prove that in a simple graph, every walk between a pair of vertices, u, v, contains a path between u and v. Recall that a walk is a list of the form v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k, in which e_i has endpoints v_{i-1} and v_i.

(B) Prove that a graph is connected if and only if for every partition of its vertices into two nonempty sets, there exists an edge that has endpoints in both sets.

Exercise 32.1.5 (Chessboard) [10 Points]

Consider a $2^n \times 2^n$ chessboard with one (arbitrarily chosen) square removed, as in the following picture (for $n = 3$):



Prove that any such chessboard can be tiled without gaps or overlaps by L-shapes consisting of 3 squares each.

Exercise 32.1.6 (Coloring) [10 Points]

(A) **[5 Points]** Let T_1, T_2 and T_3 be three trees defined over the set of vertices $\{v_1, \ldots, v_n\}$. Prove that the graph $G = T_1 \cup T_2 \cup T_3$ is colorable using six colors (e is an edge of G if and only if it is an edge in one of trees T_1, T_2 and T_3).

(B) **[5 Points]** Describe an efficient algorithm for computing this coloring. What is the running time of your algorithm?

Exercise 32.1.7 (Binary trees and codes.) Professor George O'Jungle has a favorite 26-node binary tree, whose nodes are labeled by letters of the alphabet. The preorder and postorder sequences of nodes are as follows:

preorder: `M N H C R S K W T G D X I Y A J P O E Z V B U L Q F`
postorder: `C W T K S G R H D N A O E P J Y Z I B Q L F U V X M`

Draw Professor O'Jungle's binary tree, and give the in-order sequence of nodes.

32.2 Recurrences

Exercise 32.2.1 (Recurrences.) [20 Points]

Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. More exact solutions are better.

(A) **[2 Points]** $A(n) = A\left(\sqrt{n}/3 + \lfloor \log n \rfloor\right) + n$

(B) **[2 Points]** $B(n) = \min_{0<k<n} (3 + B(k) + B(n-k))$.

(C) **[2 Points]** $C(n) = 3C(\lceil n/2 \rceil - 5) + n/\log n$

(D) **[2 Points]** $D(n) = \frac{n}{n-3} D(n-1) + 1$

(E) **[2 Points]** $E(n) = E(\lfloor 3n/4 \rfloor) + \sqrt{n}$

(F) **[2 Points]** $F(n) = F(\lfloor \log n \rfloor) + \log n$ (Hard)

(G) **[2 Points]** $G(n) = n + \left\lfloor \sqrt{n} \right\rfloor \cdot G(\left\lfloor \sqrt{n} \right\rfloor)$

(H) **[2 Points]** $H(n) = \log(H(n-1)) + 1$

(I) **[2 Points]** $I(n) = 5I\left(\left\lfloor \sqrt{n} \right\rfloor\right) + 1$

(J) **[2 Points]** $J(n) = 3J(n/4) + 1$

Exercise 32.2.2 (Recurrences II) [20 Points]

Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. More exact solutions are better.

(A) **[1 Points]** $A(n) = A(n/3 + 5 + \lfloor \log n \rfloor) + n \log\log n$

(B) **[1 Points]** $B(n) = \min_{0<k<n} (3 + B(k) + B(n-k))$.

(C) **[1 Points]** $C(n) = 3C(\lceil n/2 \rceil - 5) + n/\log n$

(D) **[1 Points]** $D(n) = \frac{n}{n-5} D(n-1) + 1$

(E) **[1 Points]** $E(n) = E(\lfloor 3n/4 \rfloor) + 1/\sqrt{n}$

(F) **[1 Points]** $F(n) = F(\left\lfloor \log^2 n \right\rfloor) + \log n$ (Hard)

(G) **[1 Points]** $G(n) = n + 7\sqrt{n} \cdot G(\left\lfloor \sqrt{n} \right\rfloor)$

(H) **[1 Points]** $H(n) = \log^2(H(n-1)) + 1$

(I) **[1 Points]** $I(n) = I\left(\left\lfloor n^{1/4} \right\rfloor\right) + 1$

(J) **[1 Points]** $J(n) = J(n - \lfloor n/\log n \rfloor) + 1$

Exercise 32.2.3 (Recurrences III) [20 Points]

Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied.

(A) $A(n) = A(n/2) + n$

(B) $B(n) = 2B(n/2) + n$

(C) $C(n) = n + \frac{1}{2}(C(n-1) + C(3n/4))$

(D) $D(n) = \max_{n/3<k<2n/3} (D(k) + D(n-k) + n)$ (Hard)

(E) $E(n) = 2E(n/2) + n/\lg n$ (Hard)

(F) $F(n) = \frac{F(n-1)}{F(n-2)}$, where $F(1) = 1$ and $F(2) = 2$. (Hard)

(G) $G(n) = G(n/2) + G(n/4) + G(n/6) + G(n/12) + n$ [Hint: $\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{12} = 1$.] (Hard)

(H) $H(n) = n + \sqrt{n} \cdot H(\sqrt{n})$ (Hard)

(I) $I(n) = (n-1)(I(n-1) + I(n-2))$, where $F(0) = F(1) = 1$ (Hard)

(J) $J(n) = 8J(n-1) - 15J(n-2) + 1$

Exercise 32.2.4 (Evaluate summations.) [10 Points] Evaluate the following summations; simplify your answers as much as possible. Significant partial credit will be given for answers in the form $\Theta(f(n))$ for some recognizable function $f(n)$.

(A) **[2 Points]** $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i}\frac{1}{i}$

(Hard)

(B) **[2 Points]** $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i}\frac{1}{j}$

(C) **[2 Points]** $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i}\frac{1}{k}$

(D) **[2 Points]** $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}\frac{1}{k}$

(E) **[2 Points]** $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}\frac{1}{j \cdot k}$

Exercise 32.2.5 (Simplify binary formulas.) This problem asks you to simplify some recursively defined boolean formulas as much as possible. In each case, prove that your answer is correct. Each proof can be just a few sentences long, but it must be a *proof.*

(A) Suppose $\alpha_0 = p$, $\alpha_1 = q$, and $\alpha_n = (\alpha_{n-2} \wedge \alpha_{n-1})$ for all $n \geq 2$. Simplify α_n as much as possible. *[Hint: What is α_5?]*

(B) Suppose $\beta_0 = p$, $\beta_1 = q$, and $\beta_n = (\beta_{n-2} \Leftrightarrow \beta_{n-1})$ for all $n \geq 2$. Simplify β_n as much as possible. *[Hint: What is β_5?]*

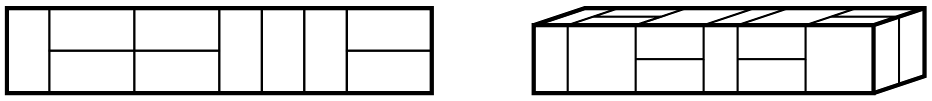
(C) Suppose $\gamma_0 = p$, $\gamma_1 = q$, and $\gamma_n = (\gamma_{n-2} \Rightarrow \gamma_{n-1})$ for all $n \geq 2$. Simplify γ_n as much as possible. *[Hint: What is γ_5?]*

(D) Suppose $\delta_0 = p$, $\delta_1 = q$, and $\delta_n = (\delta_{n-2} \bowtie \delta_{n-1})$ for all $n \geq 2$, where $\bowtie$ is some boolean function with two arguments. Find a boolean function $\bowtie$ such that $\delta_n = \delta_m$ if and only if $n - m$ is a multiple of 4. *[Hint: There is only one such function.]*

32.3 Counting

Exercise 32.3.1 (Counting dominos) (A) A *domino* is a 2×1 or 1×2 rectangle. How many different ways are there to completely fill a $2 \times n$ rectangle with n dominos? Set up a recurrence relation and give an *exact* closed-form solution.

(B) A *slab* is a three-dimensional box with dimensions $1 \times 2 \times 2$, $2 \times 1 \times 2$, or $2 \times 2 \times 1$. How many different ways are there to fill a $2 \times 2 \times n$ box with n slabs? Set up a recurrence relation and give an *exact* closed-form solution.



A 2×10 rectangle filled with ten dominos, and a $2 \times 2 \times 10$ box filled with ten slabs.

32.4 *O* notation and friends

Exercise 32.4.1 (Sorting functions) [20 Points]

Sort the following 25 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice.

$n^{4.5} - (n-1)^{4.5}$	n	$n^{2.1}$	$\lg^*(n/8)$	$1 + \lg\lg\lg n$
$\cos n + 2$	$\lg(\lg^* n)$	$(\lg n)!$	$(\lg^* n)^{\lg n}$	n^5
$\lg^* 2^{2^{2^n}}$	$2^{\lg n}$	$\sqrt{n}^e$	$\sum_{i=1}^n i$	$\sum_{i=1}^n i^2$
$n^{7/(2n)}$	$n^{3/(2\lg n)}$	$12 + \lfloor \lg\lg(n) \rfloor$	$(\lg(2+n))^{\lg n}$	$\left(1 + \frac{1}{154}\right)^{15n}$
$n^{1/\lg\lg n}$	$n^{\lg\lg n}$	$\lg^{(201)} n$	$n^{1/125}$	$n(\lg n)^4$

To simplify notation, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$ and $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the functions n^2, n, $\binom{n}{2}$, n^3 could be sorted either as $n \ll n^2 \equiv \binom{n}{2} \ll n^3$ or as $n \ll \binom{n}{2} \equiv n^2 \ll n^3$. [**Hint:** When considering two functions $f(\cdot)$ and $g(\cdot)$ it is sometime useful to consider the functions $\ln f(\cdot)$ and $\ln g(\cdot)$.]

Exercise 32.4.2 (Sorting functions II) [20 Points]

Sort the following 25 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice.

$n^{5.5} - (n-1)^{5.5}$	n	$n^{2.2}$	$\lg^*(n/7)$	$1 + \lg\lg n$
$\cos n + 2$	$\lg(\lg^* n)$	$\lg(n!)$	$(\lg^* n)^{\lg n}$	n^4
$\lg^* 2^{2^n}$	$2^{\lg^* n}$	$e^{\sqrt{n}}$	$\sum_{i=1}^n \frac{1}{i}$	$\sum_{i=1}^n \frac{1}{i^2}$
$n^{3/(2n)}$	$n^{3/(2\lg n)}$	$\lfloor \lg\lg(n!) \rfloor$	$(\lg(7+n))^{\lg n}$	$\left(1 + \frac{1}{154}\right)^{154n}$
$n^{1/\lg\lg n}$	$n^{\lg\lg n}$	$\lg^{(200)} n$	$n^{1/1234}$	$n(\lg n)^3$

To simplify notation, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$ and $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the functions n^2, n, $\binom{n}{2}$, n^3 could be sorted either as $n \ll n^2 \equiv \binom{n}{2} \ll n^3$ or as $n \ll \binom{n}{2} \equiv n^2 \ll n^3$.

Exercise 32.4.3 (*O* notation revisited.) [10 Points]

(A) Let $f_i(n)$ be a sequence of functions, such that for every i, $f_i(n) = o(\sqrt{n})$ (namely, $\lim_{n\to\infty} \frac{f_i(n)}{\sqrt{n}} = 0$). Let $g(n) = \sum_{i=1}^{n} f_i(n)$. Prove or disprove: $g(n) = o(n^{3/2})$.

(B) If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$. Prove or disprove:

- $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$
- $f_1(n) * f_2(n) = O(g_1(n) * g_2(n))$
- $f_1(n)^{f_2(n)} = O(g_1(n)^{g_2(n)})$

Exercise 32.4.4 (Some proofs required.) (A) Prove that $2^{\lceil \lg n \rceil + \lfloor \lg n \rfloor} = \Theta(n^2)$.

(B) Prove or disprove: $2^{\lfloor \lg n \rfloor} = \Theta(2^{\lceil \lg n \rceil})$.

(C) Prove or disprove: $2^{2^{\lfloor \lg \lg n \rfloor}} = \Theta(2^{2^{\lceil \lg \lg n \rceil}})$.

(D) Prove or disprove: If $f(n) = O(g(n))$, then $\log(f(n)) = O(\log(g(n)))$.

(E) Prove or disprove: If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$.
(Hard)

(F) Prove that $\log^k n = o(n^{1/k})$ for any positive integer k.

32.5 Probability

Exercise 32.5.1 (Balls and boxes.) [20 Points] There are n balls (numbered from 1 to n) and n boxes (numbered from 1 to n). We put each ball in a randomly selected box.

(A) **[4 Points]** A box may contain more than one ball. Suppose X is the number on the box that has the smallest number among all nonempty boxes. What is the expectation of X?

(B) **[4 Points]** What is the expected number of bins that have exactly one ball in them? (Hint: Compute the probability of a specific bin to contain exactly one ball and then use some properties of expectation.)

(C) **[8 Points]** We put the balls into the boxes in such a way that there is exactly one ball in each box. If the number written on a ball is the same as the number written on the box containing the ball, we say there is a match. What is the expected number of matches?

(D) **[4 Points]** What is the probability that there are exactly k matches? ($1 \le k < n$)

[Hint: If you have to appeal to “intuition” or “common sense”, your answers are probably wrong!]

Exercise 32.5.2 (Idiotic Sort) [20 Points]

There is an array A with n unsorted distinct numbers in it. IdioticSort(A) sorts the array using an iterative algorithm. In each iteration, it picks randomly (and uniformly) two indices i, j in the ranges $\{1, \ldots, n\}$. Next, if $A[\min(i, j)] > A[\max(i, j)]$ it swaps $A[i]$ and $A[j]$. The algorithm magically stop once the array is sorted.

(A) **[5 Points]** Prove that after (at most) $n!$ swaps performed by the algorithm, the array A is sorted.

(B) **[5 Points]** Prove that after at most (say) $6n^3$ swaps performed by the algorithm, the array A is sorted. (There might be an easy solution, but I don't see it.)

(C) **[5 Points]** Prove that if A is not sorted, than the probability for a swap in the next iteration is at least $\geq 2/n^2$.

(D) **[5 Points]** Prove that if A is not sorted, then the expected number of iterations till the next swap is $\leq n^2/2$. [Hint: use geometric random variable.]

(E) **[5 Points]** Prove that the expected number of iterations performed by the algorithm is $O(n^5)$. [Hint: Use linearity of expectation.]

Exercise 32.5.3 (Random walk.) [10 Points]

A *random walk* is a walk on a graph G, generated by starting from a vertex $v_0 = v \in V(G)$, and in the i-th stage, for $i > 0$, randomly selecting one of the neighbors of v_{i-1} and setting v_i to be this vertex. A walk $v_0, v_1, \ldots, v_m$ is of length m.

(A) For a vertex $u \in V(G)$, let $P_u(m, v)$ be the probability that a random walk of length m, starting from u, visits v (i.e., $v_i = v$ for some i).

Prove that a graph G with n vertices is connected, if and only if, for any two vertices $u, v \in V(G)$, we have $P_u(n-1, v) > 0$.

(B) Prove that a graph G with n vertices is connected if and only if for any pair of vertices $u, v \in V(G)$, we have $\lim_{m\to\infty} P_u(m, v) = 1$.

Exercise 32.5.4 (Random Elections.) [10 Points]

You are in a shop trying to buy green tea. There n different types of green tea that you are considering: $T_1, \ldots, T_n$. You have a coin, and you decide to randomly choose one of them using random coin flips. Because of the different prices of the different teas, you decide that you want to choose the ith tea with probability p_i (of course, $\sum_{i=1}^{n} p_i = 1$).

Describe an algorithm that chooses a tea according to this distribution, using only coin flips. Compute the expected number of coin flips your algorithm uses. (Your algorithm should minimize the number of coin flips it uses, since if you flip coins too many times in the shop, you might be arrested.)

Exercise 32.5.5 (Runs?) [10 Points]

We toss a fair coin n times. What is the expected number of "runs"? Runs are consecutive tosses with the same result. For example, the toss sequence HHHTTHTH has 5 runs.

Exercise 32.5.6 (A card game.) Penn and Teller have a special deck of fifty-two cards, with no face cards and nothing but clubs—the ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, . . . , 52 of clubs. (They're big cards.) Penn shuffles the deck until each of the 52! possible orderings of the cards is equally likely. He then takes cards one at a time from the top of the deck and gives them to Teller, stopping as soon as he gives Teller the five of clubs.

(A) On average, how many cards does Penn give Teller?

(B) On average, what is the smallest-numbered card that Penn gives Teller? (Hard)

(C) On average, what is the largest-numbered card that Penn gives Teller?

[Hint: Solve for an n-card deck and then set $n = 52$.] In each case, give *exact* answers and prove that they are correct. If you have to appeal to "intuition" or "common sense", your answers are probably wrong!

Exercise 32.5.7 (Alice and Bob) Alice and Bob each have a fair n-sided die. Alice rolls her die once. Bob then repeatedly throws his die until he rolls a number at least as big as the number Alice rolled. Each time Bob rolls, he pays Alice \$1. (For example, if Alice rolls a 5, and Bob rolls a 4, then a 3, then a 1, then a 5, the game ends and Alice gets \$4. If Alice rolls a 1, then no matter what Bob rolls, the game will end immediately, and Alice will get \$1.)

Exactly how much money does Alice expect to win at this game? Prove that your answer is correct. If you have to appeal to 'intuition' or 'common sense', your answer is probably wrong!

32.6 Basic data-structures and algorithms

Exercise 32.6.1 (Storing temperatures.) [10 Points]

Describe a data structure that supports storing temperatures. The operations on the data structure are as follows:

`Insert`(t, d) — Insert the temperature t that was measured on date d. Each temperature is a real number between -100 and 150. For example, `insert`(22, "1/20/03").

`Average`(d_1, d_2) report what is the average of all temperatures that were measured between date d_1 and date d_2.

Each operation should take time $O(\log n)$, where n is the number of dates stored in the data structure. You can assume that a date is just an integer which specifies the number of days since the first of January 1970.

Exercise 32.6.2 (Binary search tree modifications.) [10 Points]

Suppose we have a binary search tree. You perform a long sequence of operations on the binary tree (insertion, deletions, searches, etc), and the maximum depth of the tree during those operations is at most h.

Modify the binary search tree T so that it supports the following operations. Implementing some of those operations would require you to modify the information stored in each node of the tree, and the way insertions/deletions are being handled in the tree. For each of the following, describe separately the changes made in detail, and the algorithms for answering those queries. (Note, that under the modified version of the binary search tree, insertion and deletion should still take $O(h)$ time, where h is the maximum height of the tree during all the execution of the algorithm.)

(A) **[2 Points]** Find the smallest element stored in T in $O(h)$ time.

(B) **[2 Points]** Given a query k, find the k-th smallest element stored in T in $O(h)$ time.

(C) **[3 Points]** Given a query $[a, b]$, find the number of elements stored in T with their values being in the range $[a, b]$, in $O(h)$ time.

(D) **[3 Points]** Given a query $[a, b]$, report (i.e., printout) all the elements stored in T in the range $[a, b]$, in $O(h + u)$ time, where u is the number of elements printed out.

Exercise 32.6.3 (Euclid revisited.) [10 Points]

Prove that for any nonnegative parameters a and b, the following algorithms terminate and produce identical output. Also, provide bounds on the running times of those algorithms. Can you imagine any reason why WeirdEuclid would be preferable to FastEuclid?

```
SlowEuclid($a, b$) :
  if $b > a$
      return SlowEuclid($b, a$)
  else if $b = 0$
      return $a$
  else
      return SlowEuclid($b, a - b$)
```

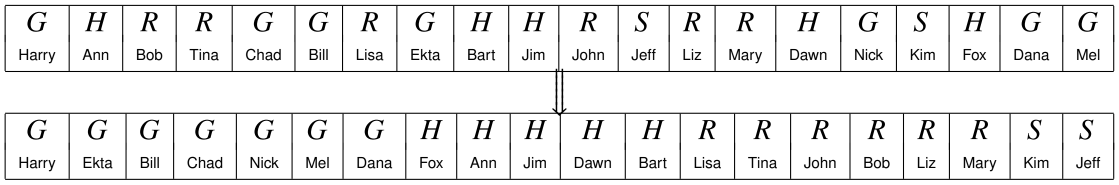
```
FastEuclid($a, b$) :
  if $b = 0$
      return $a$
  else
      return FastEuclid($b, a \bmod b$)
```

```
WeirdEuclid($a, b$) :
  if $b = 0$
      return $a$
  if $a = 0$
      return $b$
  if $a$ is even and $b$ is even
      return $2*$WeirdEuclid($a/2, b/2$)
  if $a$ is even and $b$ is odd
      return WeirdEuclid($a/2, b$)
  if $a$ is odd and $b$ is even
      return WeirdEuclid($a, b/2$)
  if $b > a$
      return WeirdEuclid($b - a, a$)
  else
      return WeirdEuclid($a - b, b$)
```

Exercise 32.6.4 (This despicable sorting hat trick.) Every year, upon their arrival at Hogwarts School of Witchcraft and Wizardry, new students are sorted into one of four houses (Gryffindor, Hufflepuff, Ravenclaw, or Slytherin) by the Hogwarts Sorting Hat. The student puts the Hat on their head, and the Hat tells the student which house they will join. This year, a failed experiment by Fred and George Weasley filled almost all of Hogwarts with sticky brown goo, mere moments before the annual Sorting. As a result, the Sorting had to take place in the basement hallways, where there was so little room to move that the students had to stand in a long line.

After everyone learned what house they were in, the students tried to group together by house, but there was too little room in the hallway for more than one student to move at a time. Fortunately, the Sorting Hat took CS Course many years ago, so it knew how to group the students as quickly as possible. What method did the Sorting Hat use?

(A) More formally, you are given an array of n items, where each item has one of four possible values, possibly with a pointer to some additional data. Describe an algorithm[2] that rearranges the items into four clusters in $O(n)$ time using only $O(1)$ extra space.

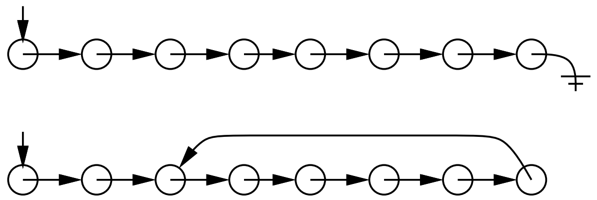

G H R R G G R G H H R S R R H G S H G G
Harry Ann Bob Tina Chad Bill Lisa Ekta Bart Jim John Jeff Liz Mary Dawn Nick Kim Fox Dana Mel
G G G G G G G H H H H H R R R R R R S S
Harry Ekta Bill Chad Nick Mel Dana Fox Ann Jim Dawn Bart Lisa Tina John Bob Liz Mary Kim Jeff

(B) Describe an algorithm for the case where there are k possible values (i.e., $1, 2, \ldots, k$) that rearranges the items using only $O(\log k)$ extra space. How fast is your algorithm? (A faster algorithm would get more credit)

(C) Describe a faster algorithm (if possible) for the case when $O(k)$ extra space is allowed. How fast is your algorithm?

(D) (Hard)Provide a fast algorithm that uses only $O(1)$ additional space for the case where there are k possible values.

Exercise 32.6.5 (Snake or shake?) Suppose you have a pointer to the head of singly linked list. Normally, each node in the list only has a pointer to the next element, and the last node's pointer is NULL. Unfortunately, your list might have been corrupted by a bug in somebody else's code[3], so that the last node has a pointer back to some other node in the list instead.



Top: A standard linked list. Bottom: A corrupted linked list.

Describe an algorithm that determines whether the linked list is corrupted or not. Your algorithm must not modify the list. For full credit, your algorithm should run in $O(n)$ time, where n is the number of nodes in the list, and use $O(1)$ extra space (not counting the list itself).

[2]Since you've read the Homework Instructions, you know what the phrase 'describe an algorithm' means. Right?

[3]After all, *your* code is always completely 100% bug-free. Isn't that right, Mr. Gates?

32.7 General proof thingies

Exercise 32.7.1 (Cornification) [20 Points]

Cornification - Conversion into, or formation of, horn; a becoming like horn. Source: Webster's Revised Unabridged Dictionary.

During the sweetcorn festival in Urbana, you had been kidnapped by an extreme anti corn organization called Al Corona. To punish you, they give you several sacks with a total of $(n+1)n/2$ cobs of corn in them, and an infinite supply of empty sacks. Next, they ask you to play the following game: At every point in time, you take a cob from every non-empty sack, and you put this set of cobs into a new sack. The game terminates when you have n non-empty sacks, with the ith sack having i cobs in it, for $i = 1, \ldots, n$.

For example, if we started with $\{1, 5\}$ (i.e., one sack has 1 cob, the other 5), we would have the following sequence of steps: $\{2, 4\}$, $\{1, 2, 3\}$ and the game ends.

(A) **[5 Points]** Prove that the game terminates if you start from a configuration where all the cobs are in a single sack.

(B) **[5 Points]** Provide a bound, as tight as possible, on the number of steps in the game till it terminates in the case where you start with a single sack.

(C) **[5 Points]** (hard) Prove that the game terminates if you start from an arbitrary configuration where the cobs might be in several sacks.

(D) **[5 Points]** Provide a bound, as tight as possible, on the number of steps in the game till it terminates in the general case.

Exercise 32.7.2 (Fibonacci numbers.) Recall the standard recursive definition of the Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. Prove the following identities for all positive integers n and m.

(A) F_n is even if and only if n is divisible by 3.

(B) $\sum_{i=0}^{n} F_i = F_{n+2} - 1$

(C) $F_n^2 - F_{n+1}F_{n-1} = (-1)^{n+1}$ (Really Hard)

(D) If n is an integer multiple of m, then F_n is an integer multiple of F_m.

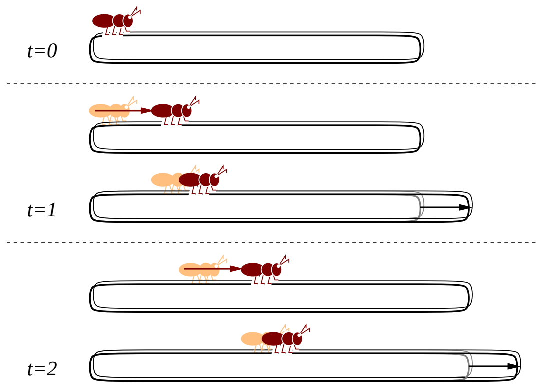
Exercise 32.7.3 (Some binomial identities.) (A) Prove the following identity by induction:

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}\binom{n}{n-k}.$$

(B) Give a non-inductive combinatorial proof of the same identity, by showing that the two sides of the equation count exactly the same thing in two different ways. There is a correct one-sentence proof.

32.8 Miscellaneous

Exercise 32.8.1 (A walking ant.) (Hard)An ant is walking along a rubber band, starting at the left end. Once every second, the ant walks one inch to the right, and then you make the rubber band one inch longer by pulling on the right end. The rubber band stretches uniformly, so stretching the rubber band also pulls the ant to the right. The initial length of the rubber band is n inches, so after t seconds, the rubber band is $n + t$ inches long.



Every second, the ant walks an inch, and then the rubber band is stretched an inch longer.

(A) How far has the ant moved after t seconds, as a function of n and t? Set up a recurrence and (for full credit) give an *exact* closed-form solution. [Hint: What *fraction* of the rubber band's length has the ant walked?]

(B) How long does it take the ant to get to the right end of the rubber band? For full credit, give an answer of the form $f(n) + \Theta(1)$ for some explicit function $f(n)$.