Homework 5
Fundamental Algorithms, Spring 2003, Professor Yap

Due: Wed May 7, in class

INSTRUCTIONS:

- Please read the questions carefully. When in doubt, please ask.

---

1. (20 Points) Recursive Dynamic Programming
Equation (1) Lecture VII (p.12) describes the Dynamic Programming solution for the optimal triangulation problem. The "bottom-up" implementation of this equation as a triply-nested for-loop was given in page 14. In this question, we want to solve this "top-down", by using a recursive algorithm. We assume that the weight $W(i,j,k)$ is easily computed for any $1 \le i < j < k \le n$.
(i) Briefly explain how a naive recursive algorithm would be exponential.
(ii) Describe an efficient recursive algorithm for optimal triangulation. You will need to use some global data structure for sharing information across subproblems.
(iii) Briefly analyze the complexity of your solution.
(iv) Does your algorithm ever run faster than the bottom-up implementation? Can you make it run faster on some inputs?

2. (20 Points) String Alignment Problem
Recall the string edit problem, where the operations are Insert, Delete and Replace. Given two strings X, Y, we want the minimum "cost" of modifying X so that the two strings becomes identical. Here is our definition of "cost":
(a) Each deletion or insertion costs 2 units.
(b) Each replacement costs 1 unit.
(c) Each "original" **match** involving a character from X and the same character in Y costs -2 units. What we mean by "original" match will be clear in an example. Note that this cost occurs in the absence of any operations!

   In contrast, in the original string edit problem, matches have no cost while each of the other operations costs 1 unit. The minimum costs of such a sequence of operations (or non-operations) is called the **alignment cost** for X, Y and is denoted $A(X,Y)$. For instance, $X = cga$ and $Y = acaat$. We can first insert a in the front of X to get $X' = acga$ (costs 2), and delete t from Y to get $Y' = acaa$. To indicate the various copies of a letter, we use subscripts: $X' = a_1cga_2$ and $Y' = a_3ca_4a_5$. Now we make the obvious one-one correspondence between X' and Y': we get matches at positions $1, 2, 4$. The matches at positions 2 and 4 costs -2 units each. However, the match at position 1 (between a_1 with a_3) is a result of inserting a_1 and so it has already been counted. In other words, it is not an "original match". Since there is no match at 3, we need to replace g by an a to get a match – this replacement costs 1 unit. This shows that $A(X,Y) \le 2 + 2 - 2 - 2 + 1 = 1$. [Can you show a lower cost alignment of X, Y?]

   To summarize, here is how we perform a single alignment of X and Y: first perform a sequence of insertions and deletions until the strings have the same length. Then align the result in the obvious way, and find all the original matches, as well as perform any replacements as needed. Add up the cost of deletions/insertions, original matches and replacements. This problem arise in DNA sequencing in computational biology.

   (i) Give the recursive formula for $A(X,Y)$ and justify it.
(ii) Compute $A(X,Y)$ where X, Y are the strings `AATTCCCGA` and `GCATATT`. You must organize this computation systematically as in the LCS problem.

3. (20 Points) Probabilistic Counters
Recall the counter problem where, given a binary counter C which is initially 0, you can perform the operation `inc`(C) to increments its value by 1. Now we want to do **probabilistic counting**: each time you call `inc`(C), it will flip a fair coin. If heads, the value of C is incremented and otherwise the

value of C is unchanged. Now, at any moment you could call `look`(C), which will return *twice* the current value of C. Let X_m be the value of `look`(C) after you have made m calls to `inc`(C).
(a) Note that X_m is a random variable. What is the sample space Ω here?
(b) Let $P_m(i)$ be the probability that `look`$(C) = i$ after m `inc`'s. State a recurrence equation for $P_m(i)$ involving $P_{m-1}(i)$ and $P_{m-1}(i-1)$.
(c) Give the exact formula for $P_m(i)$ using binomial coefficients. HINT: you can either use the model in (a) to give a direct answer, or you can try to solve the recurrence of (b). You may recall that binomial identity $\binom{m}{i} = \binom{m-1}{i} + \binom{m-1}{i-1}$.
(d) In probabilistic counting we are interested in the *expected* value of `look`(C), namely $E[X_m]$. What is the expected value of X_m? HINT: express $E[X_m]$ using $P_m(i)$ and do some simple manipulation involving binomial coefficients. If you do not see what is coming out, try small examples like $m = 2, 3$ to see what the answer is.

[NOTE: The expected value of X_m can be odd even when the actual value returned is always even. Using a generalization of these ideas, you can probabilistically count to 2^{2^n} with an n-bit counter.]

4. (20 Points) Hashing

(a) Compute the sequence $\{\alpha\}, \{2\alpha\}, \ldots, \{n\alpha\}$ for $n = 10$ and $\alpha = \phi$ (= the golden ratio $(1+\sqrt{5})/2 = 1.618\ldots$). You may compute to just 4 decimal positions using any means you like.
(b) Let

$$\ell_0 > \ell_1 > \ell_2 > \cdots$$

be the new lengths of subsegments, in order of their appearance as we insert the points $\{n\phi\}$ (for $n = 0, 1, 2\ldots$) into the unit interval. For instance, $\ell_0 = 1, \ell_1 = 0.61803, \ell_2 = 0.38197$. Compute ℓ_i for $i = 0, \ldots, 10$.
(c) Using the multiplication method with $\alpha = \phi$, please insert the following set of 16 keys into a table of size $m = 10$. Treat the keys as integers by treating the letters `A, B, ..., Z` as $1, 2, \ldots, 26$, with the rightmost position having a value of 1, the next position with value 26, the third with value $26^2 = 676$, etc. Thus `AND` represents the integer $(1 \times 26^2) + (15 \times 26) + (4 \times 1) = 1070$. This is sometimes called the **26-adic** notation. To resolve collision, use separate chaining.

```
AND, ARE, AS, AT, BE, BOY, BUT, BY, FOR, HAD,
HER, HIS, HIM, IN, IS, IT
```

We just want you to display the results of your final hashing data structure.
(d) Use the division method on the same set of keys as (c), but with $m = 17$.

5. (20 Points) NP-Completeness
We guide you through Exercise 6.2 in Lecture XXX on NP-Completeness. The problem L is to recognize whether a given bigraph G is "triangular" or not. To show that L is Karp-reducible to SAT, you need to construct a Boolean formula $\phi(G)$ such that G is triangular iff $\phi(G) \in SAT$. Moreover, this construction must be done in polynomial time.
(i) If $G = (V, E)$ and $|V|$ is not divisible by 3 then there is no solution. What would you output as $\phi(G)$ in this case?
(ii) Suppose $|V| = 3m$. So our goal is to form m disjoint triangles from the vertices of G. Introduce the Boolean variable x_{ij} which corresponds to the proposition "Node i is in the jth Triangle". Here, $i \in V$ and $j = 1, \ldots, m$. Using these variables, you construct a Boolean formula $F_1(i)$ that is true iff i is in at least one of the m triangles?
(iii) Similarly, construct $F_2(i)$ that is true iff i is in at most one triangle.
(iii) Construct a formula $F_3(j)$ that is true iff the jth triangle has at least three nodes.
(iv) Construct a formula $F_4(j)$ that is true iff the jth triangle has at most three nodes.
(v) Construct a formula $F_5(j)$ that is true iff each pair of vertices in the jth triangle has an edge in the graph G. [NOTE: this is the first time you are actually using specific information about the edges of G. You know G since it is in the input.]
(vi) Using the above formulas, describe a formula $F(G)$ that is true iff G is triangular. You must prove

this claim about $F(G)$.
(vii) Conclude that L is Karp-reducible to SAT.

THE REMAINING QUESTIONS CARRIES NO CREDIT. BUT YOU ARE ENCOURAGED TO GO OVER THEM. IF YOU HAVE NO TIME, AT LEAST THINK TRY TO UNDERSTAND THE ISSUES AND HAVE A STRATEGY TO SOLVE THEM.

6. (0 Points)
(a) Suppose you have a random number generator, which is a function $rand()$ that returns a real number r in the range $0 \leq r < 1$ with "uniform" probability (this means that for any $0 \leq a < b \leq 1$, the probability of r lying in $[a, b]$ is $b - a$). Given any n, how do you generate an integer in the range $\{0, 1, \ldots, n-1\}$ with equal probability?
(b) Given an array $A[0..n]$ of numbers ($A[i]$ is some number x_i). How do you compute a random permutation of these numbers?

7. (0 Points)
To understand Coalesced Chaining, we ask you to provide the algorithms for LookUp(key k), Insert(key k, data d), and Delete(key k) for this form of hashing. As usual, we assume a hash function $h : U \to \mathbb{Z}_m$. The hash table is denote $T[0..m-1]$ where the ith entry is $T[i]$. We assume that $T[i]$ has three fields, $T[i]$.`Key`, $T[i]$.`Data` and $T[i]$.`next`. The value of $T[i]$.`Key` is either a key (an element of U) or the special values EMPTY or DELETED. The value of $T[i]$.`next` is either an element of $\mathbb{Z}_m$ or -1. Provide the algorithmic details for the three dictionary operations: LookUp, Insert and Delete.

8. (0 Points)
Describe the algorithmic details of our offline Quicksort algorithm in the notes. You must make very explicit choices for the data structures (how the input is represented and how whether you are using linked lists, etc).