Problem Set 5

Due Date: March 14, 2000.

Self-study:

Please review Rosen Section 2.6, on matrices.

Self-study problems: 3, 5, 11, 13, 15, 19, 23, 29, 30, 31

Reading:

Notes on Computational Processes:
Installment 3

Rosen, *Discrete Mathematics and Its Applications*:
Sequences and Summations: Section 1.7;

6.042 Fall 97 Lecture notes (available from course page):
Lecture 8

Problems:

Problem 1 Suppose you were about to enter college today and a college loan officer offered you the following deal: \$25,000 at the start of each year for four years to pay for your college tuition and an option of choosing one of the following repayment plans:

Plan A: Wait four years, then repay $\$20,000$ at the start of each year for the next ten years.

Plan B: Wait five years, then repay $\$30,000$ at the start of each year for the next five years.

Suppose the annual interest rate is currently 7% and does not change in the future.

a) Which repayment plan would you choose?

b) What is the loan officer's effective profit (in today's dollars) on the loan?

Problem 2 Consider the following summation:

$$\sum_{k=2}^{n} \frac{1}{k(k-1)}$$

(a) Find a closed form solution for this summation.

Hint: Try using partial fractions to break $\frac{1}{k(k-1)}$ into a difference of two terms.

(b) Use induction to verify your closed form solution (i.e., prove that it is correct).

Problem 3 Find closed-form expressions for the following sums:

(a)

$$\sum_{k=0}^{n} \frac{8^k - 5^k}{9^k}$$

(b)

$$\sum_{i=1}^{n} \sum_{j=0}^{\infty} i^{7/16} \cdot \left(1 - \frac{1}{4i^{9/16}}\right)^j$$

Problem 4 *Note: This problem is only worth one point. Give it some thought but don't exhaust yourself on it.*

Consider the sum of the following sequence:

$$1 - 1 + \frac{1}{2} - \frac{1}{2} + \frac{1}{3} - \frac{1}{3} + \frac{1}{4} - \frac{1}{4} + \frac{1}{5} - \frac{1}{5} + \ldots$$

To sum it, suppose we rearrange the terms and write it as:

$$\begin{array}{lc} (1 + \frac{1}{2} - 1) & + \\ (\frac{1}{3} + \frac{1}{4} - \frac{1}{2}) & + \\ (\frac{1}{5} + \frac{1}{6} - \frac{1}{3}) & + \\ (\frac{1}{7} + \frac{1}{8} - \frac{1}{4}) & + \\ \ldots & \end{array}$$

which we can write as a summation as:

$$a = \sum_{n=1}^{\infty} \left(\frac{1}{2n-1} + \frac{1}{2n} - \frac{1}{n} \right) = \sum_{n=1}^{\infty} \frac{1}{2n(2n-1)}$$

On the other hand, we arrange the original summation as

$$\begin{array}{lc} (1 - 1 - \frac{1}{2}) & + \\ (\frac{1}{2} - \frac{1}{3} - \frac{1}{4}) & + \\ (\frac{1}{3} - \frac{1}{5} - \frac{1}{6}) & + \\ (\frac{1}{4} - \frac{1}{7} - \frac{1}{8}) & + \\ \ldots & \end{array}$$

$$b = \sum_{n=1}^{\infty} \left(\frac{1}{n} - \frac{1}{2n-1} - \frac{1}{2n} \right) = \sum_{n=1}^{\infty} \frac{-1}{2n(2n-1)} = -a$$

How is this possible? Choose the correct response from the options below. Be prepared to back up your answer.

a) I find the above somewhat disturbing, but I can't resolve it.

b) Don't toy with me. Your petty tricks are no match for my superior mathematical skills.

c) It's 4am Tuesday morning. I'm not even coherent enough to be disturbed.

Problem 5 The following labeled state machine *DSet* models a "dynamic set" of integers. (Remember that a set, unlike a multiset, can only contain one element of any particular object). The set starts out empty.

A client can modify or inspect the set by means of the following operations:

- **add**(x), which adds an integer x to the set.
- **remove**(x), which removes integer x from the set.
- **isin**(x), which tells if the element x is in the set.

Formally, *DSet* can be defined as a collection of the following components:

- Q: The states consist of the components:
  - *set*, a set of integers
  - $reqtype \in \{\text{"}add\text{"}, \text{"}remove\text{"}, \text{"}isin\text{"}\} \cup \{null\}$
  - $element \in \mathbb{Z} \cup \{null\}$
  - $retval \in \{\text{"}OK\text{"}, null\} \cup Boolean$
- Q_0: Initially, $set = \emptyset$, and all the other components are *null*. (Note that *null* is a special symbol, not the empty set $\emptyset$).
- L:

  Input labels:

  request-add(x), $x \in \mathbb{Z}$
  request-remove(x), $x \in \mathbb{Z}$
  request-isin(x), $x \in \mathbb{Z}$

  Internal labels:

  compute-add(x), $x \in \mathbb{Z}$
  compute-remove(x), $x \in \mathbb{Z}$
  compute-isin(x), $x \in \mathbb{Z}$

  Output labels:

  return("OK")
  return(b), $b \in Boolean$

- Transitions (δ):

  request-add(x)
  if $reqtype = null$ then
  $reqtype :=$ "add"
  $element := x$

  request-remove(x)
  if $reqtype = null$ then
  $reqtype :=$ "$remove$"
  $element := x$

request-isin(x)
if *reqtype* $=$ *null* then
  reqtype $:=$ "*isin*"
  element $:= x$

compute-add(x)
Can occur if *reqtype* $=$ "*add*", *element* $= x$, and *retval* $=$ *null*
  set $:=$ *set* $\cup \{x\}$
  retval $:=$ "*OK*"

compute-remove(x)
Can occur if *reqtype* $=$ "*remove*", *element* $= x$, and *retval* $=$ *null*
  set $:=$ *set* $- \{x\}$
  retval $:=$ "*OK*"

compute-isin(x)
Can occur if *reqtype* $=$ "*isin*", *element* $= x$, and *retval* $=$ *null*
if $x \in$ *set* then
  retval $:=$ *true*
else
  retval $:=$ *false*

return("*OK*")
Can occur if *retval* $=$ "*OK*"
  reqtype, *element*, *retval* $:=$ *null*

return(b), $b \in$ *Boolean*
Can occur if *retval* $= b$
  reqtype, *element*, *retval* $:=$ *null*

(a) Which of the following are valid traces of *DSet*? For any incorrect ones, indicate what is wrong.

1. Start with the sole element of Q_0.
   - **request-add**(7)
   - **return**("*OK*")
   - **request-isin**(7)
   - **return**(*true*)
   - **request-remove**(7)
   - **return**("*OK*")
   - **request-remove**(7)
   - **return**("*OK*")

2. Start with the sole element of Q_0.
   - **request-add**(7)
   - **return**("*OK*")
   - **request-remove**(4)

- **return**(*false*)
- **request-add**(7)
- **return**(“*OK*”)

3. Start with the sole element of Q_0.

- **request-add**(4)
- **request-add**(6)
- **return**(“*OK*”)
- **request-isin**(6)
- **request-isin**(4)
- **return**(*true*)

4. Start with the sole element of Q_0.

- **request-add**(3)
- **request-remove**(3)
- **return**(“*OK*”)
- **request-isin**(3)
- **return**(*false*)
- **request-remove**(1)
- **return**(“*OK*”)

(b) High-level implementation:

One way to implement *DSet* is to represent the set by a finite sequence of integers. The **add** operation could append the new element to the end of the sequence *whether or not it already appears in the sequence*. The **isin** operation returns the answer to whether or not any copy of the element is in the sequence. The **remove** operation removes the element from the sequence at every place that it occurs.

Define a state machine *Seq* corresponding to this implementation. In implementing the operations, you should not try to represent individual search steps that try to locate an element in the sequence. Instead, let each operation perform its main computation in one large-granularity step, which involves the entire sequence. Thus, this description will still be rather abstract.

(c) Prove that your *Seq* state machine implements the *DSet* labeled state machine; that is, prove that any trace of *Seq* is a trace of *DSet*. Remember that a trace is a listing of only the input and output labels during an execution — the externally visible behavior. You may use the Abstraction Theorem.

(d) Low-level implementation:

Give an implementation called *SeqLow* that relies on lower-level operations for performing the searches and removals (operate at the element-level).

(e) Give an informal argument that *SeqLow* implements *Seq*.

Problem 6 Consider an "abstract object" that maintains a dynamic polynomial *poly*. This polynomial has variable x, and integer coefficients. Initially, the polynomial is just the 0 polynomial (no terms). The polynomial supports operations:

- **add-term**(c, e), which adds the term cx^e to the maintained polynomial.
- **coeff**(e), which asks for the coefficient of the x^e term.

(a) Model this object by a labeled state machine ADP. The state should keep track of the polynomial (a value that is simply a mathematical polynomial — you don't have to think about how to represent it using lower-level mathematical objects like numbers, matrices, sequences, etc.) The state will also need some bookkeeping information to keep track of operations in progress. The labels should include inputs reflecting the requests and outputs reflecting the responses. Your machine should ignore requests that arrive while an operation is active.

(b) List some (4–5) traces that are exhibited by your state machine ADP. Include at least one that involves a request arriving when it is not supposed to.

(c) Describe in words an implementation of your state machine ADP. The state of this implementation should involve lower-level mathematical objects like numbers, matrices, sequences, etc.

(d) Formalize your implementation as a labeled state machine *Imp*. It should have the same external labels as your labeled state machine ADP.

(e) Argue that your implementation *Imp* is correct, in the sense that every trace of *Imp* is also a trace of ADP.

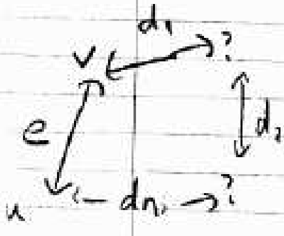
Problem 7 Attached to this problem set are some students' solutions from a previous problem set. Mark up these solutions, indicating things that are incorrect or unclear, and if possible show how the solution can be made completely correct and clear. Please refer to the 6.042 course webpage for more readable versions of the students' solutions.

(a) Problem Set 4, Problem 1

1. Prove that an edge e in a connected graph G is a cut edge if and only if no simple cycle contains e

Prove technique = Contradiction

1. If edge e in G is a cut edge, there will be no simple cycle containing e



1. Assume e is a cut edge and there is a simple cycle containing e.
2. ∃ path from u to v where $u, v \in G$ (G is connected)
3. ~~The~~ If we remove e, 1, 2 still true
   1. Let $d_1, d_2, d_3, \ldots d_n$ be the edges in a simple cycle, then there is a path just like that:
      $$e \rightleftarrows d_1 \rightleftarrows d_2 \rightleftarrows d_3 \ldots \rightleftarrows d_n \rightleftarrows e,$$
   2. Pick arbitrary edge e, which connects u to v
   3. Removing it, there is still a path from u to v, say:
      $$d_1 \to d_2 \to d_3 \to \ldots d_n$$
   4. Therefore we can construct a path from u to v in the absence of e

   QED
4. This contradicts the definition of cut edge
   1. Since e is removed, the Graph is disconnected (def of cut edge)
   2. There shouldn't exist a path from arbitrary u, v if the graph is disconnected
   3. Contradiction

QED

QED

(b) Problem Set 4, Problem 2

(2) Prove $r(u) = r(v)$, where $r(x)$ is the average of x's neighbors, for all $u, v \in V$.

1. Assume $r(u) \neq r(v)$ for all $u, v \in V$.
2. Also assume $r(v)$ is smallest member of V. (by well-ordering)
3. Then $r(v) \neq$ average of all neighbors.
   1. $r(v)$ is smallest member and $r(v) \neq r(u)$, thus $r(u) > r(v)$.
   2. If $r(u) > r(v)$ and $r(v)$ is the average of $r(u)$ and its other neighbors, one or more of the other neighbors must be $< r(v)$. (by simple arithmetic/algebra)
   3. However, no neighbor can be $< r(v)$. (by assumption and well-ordering.)
   4. QED.
4. QED. (by contradiction)