Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-1

Collaborators:

Give a linear-time algorithm that takes two sorted arrays of real numbers as input, and returns a merged list of sorted numbers. You should give your answer in pseudocode. Your answer should contain:

- A prose explanation of the algorithm.
- Psuedocode. (Be sure to review the two resources on pseudocode that were posted as readings for Week 2! I also suggest the algorithm / algorithmx package in LaTex.)

Algorithm 1 Merged list of sorted numbers from two sorted lists

```
procedure MERGE(A, B)                    ▷ A and B are sorted lists this sorts them into list c
    in: Sorted lists A,B
    out: Sorted list c, the combination of A and B
    c ← list;
    i, j ← 0;
    while i < A.size()&&j < B.size() do
        if A.get(i) == B.get(j) then
            c.add(A.get(i), B.get(j));
            i++;
            j++;
        else if A.get(i) < B.get(j) then
            c.add(A.get(i));
            i++;
        else if A.get(i) > B.get(j) then
            c.add(B.get(j))
            j++;
        end if
    end while
    if i == A.size() then
        while j < B.size() do
            c.add(B.get(j));
            j++;
        end while
    end if
    if j == B.size() then
        while i < A.size() do
            c.add(A.get(i));
            i++;
        end while
    end if
    return c;
end procedure
```

- The decrementing function for any loop or recursion.

D(x) : x = (length(A) + length(B))-(i + j) $\rightarrow \mathbb{Z}$
Loop terminates when D reaches 0.
D(x) : x = (length(A) - i) $\rightarrow \mathbb{Z}$
Loop terminates when D reaches 0.
D(x) : x = (length(B) - j) $\rightarrow \mathbb{Z}$
Loop terminates when D reaches 0.

- Justification of why the runtime is linear.

  The algorithm will go through every item in the lists exactly once since the counters i and j will increment until they hit the array size and therefore no item in the lists will be have more than O(1) spent on it. This mean that O(A.size()+B.size()) is the complexity and therefore it is run in linear time.

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-2

Collaborators:

EPI 15.4 (Generate the Power Set) gives code to compute the power set of a set (without duplicates). Present this problem and solution in your own words using pseudocode.

Algorithm 2 Power Sets

```
procedure GENERATEPOWERSETS(inputSet)                                  ▷ Startup function
    in: List of integers that is a the inputSet
    out: List of Lists of Integers that are the power sets;
    powerSet ← list;
    newList ← list;
    directedPowerSet(inputSet, 0, newList, powerSet);
    return powerSet;
end procedure
procedure DIRECTEDPOWERSET(inputSet, toBeSelected, selectedSoFar, powerSet)
    in: inputSet : the original input set, toBeSelected : the spot in inputSet that the algorithm is checking, selectedSoFar : list of spots in inputSet already checked, powerSet : list of power sets already selected
    out: None
    if toBeSelected == inputSet.size() then
        powerSet.add(selectedSoFar.asList());   ▷ Adds all of selected so far because they represent a powerSet to powerSet and ends because there is nothing left to check
        return;
    end if
    selectedSoFar.add(inputSet.get(toBeSelected));
    directedPowerSet(inputSet, toBeSelected + 1, selectedSoFar, powerSet);
    selectedSoFar.remove(selectedSoFar.size() - 1);
    directedPowerSet(inputSet, toBeSelected + 1, selectedSoFar, powerSet);
end procedure
```

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-3

Collaborators:

In EPI 15.1 (The Towers of Hanoi Problem), prove that the algorithm as presented terminates. In particular, you should give the decrementing function for the recursion.

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-4

Collaborators:

For the stock market problem discussed in class on September 6th (and in CLRS 4.1), walk through the algorithm for the following input:

$$\texttt{price} = \{3, 6, 8, 2, 1, 10, 5, 7\}.$$

BuySell({n[1]:3, n[2]:6, n[3]:8, n[4]:2}), BuySell({n[5]:1, n[6]:10, n[7]:5, n[8]:7})
BuySell({n[1]:3,n[2]: 6}) BuySell({n[3]:8, n[4]:2}), BuySell({n[5]:1, n[6]:10}), BuySell({n[7]:5, n[8]:7})
compare({n[1]:3,n[2]:6},{n[3]:2,n[4]:8},{n[1]:3,n[4]:8}) = {n[3]:2,n[4]:8}, compare({n[5]:1,n[6]:10},{n[7]:5, n[8]:7},{n[5]:1,n[8]:7
= {n[5]:1, n[6]:10}
compare({n[3]:2,n[4]:8},{n[5]:1,n[6]:10},{n[3]:2,n[6]:10}) = {n[5]:1,n[6]:10}
result = {n[5],n[6]}

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-5

Collaborators:

Prove using induction that the closed form of:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + n & n > 1 \end{cases}$$

is $O(n^2)$.

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-6

Collaborators:

What is the closed form of the following recurrence relations? Use Master's theorem to justify your answers:

1. $T(n) = 16T(n/4) + \Theta(n)$

   $a = 16, b = 4, n^2, f(n) = n, case 1$
   $\epsilon = 1$ T(n) $= \Theta(n^2)$

2. $T(n) = 2T(n/2) + n \log n$

   $a = 2, b = 2, n^1, f(n) = n \log n, case 3$
   $f(n) = \Theta(n^c), c = 2$
   $\log_2 2 < 2$ satisfies condition for case 3
   T(n) $= \Theta(n \log n)$

3. $T(n) = 6T(n/3) + n^2 \log n$

   $a = 6, b = 3, n^1.6, f(n) = n^2, case 3$
   $f(n) = \Theta(n^c), c = 2$
   $\log_3 6 < 2$ satisfies condition for case 3
   T(n) $= \Theta(n^2)$

4. $T(n) = 4T(n/2) + n^2$

   $a = 4, b = 2, n^2, f(n) = n^2, case 2$
   T(n) $= \Theta(n^2 \log n)$

5. $T(n) = 9T(n/3) + n$

   $a = 9, b = 3, n^2, f(n) = n, case 1$
   $\epsilon = 1$ T(n) $= \Theta(n^2)$

Note: we assume that $T(1) = \Theta(1)$ whenever it is not explicitly given.

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-7

Collaborators:

The skyline problem: You are waiting for the ferry across the river to get into a big city, and notice n buildings in front of you. You take a photo, and notice that each building has the silhouette of a rectangle. Suppose you represent each building as a triple (x_1, x_2, y), where the building can be seen from x_1 to x_2 horizontally and has a height of y. Let `rect(b)` be the set of points inside this rectangle (including the boundary). Let `building` be the set of n triples. Design an algorithm that takes `buildings` as input, and returns the skyline, where the skyline is a sequence of (x, y) coordinates defining $\cup_{b \in \texttt{buildings}} \texttt{rect}(b)$.

Goal is to use a divide and conquer algorithm (said in class).

Algorithm 3 Skyline Problem

```
procedure getSkyline(buildingCoord)
    in: list of skyline variables as buildingCoord.
    out: final list of coord for the skyline.
    return calculateSkyline(buildingCoord, 0, buildingCoord.size() - 1)
end procedure
procedure calculateSkyline(arr, l, h)
    in: list of coordinate as arr, the low spot in the array as l, the high spot in the array as h
    out: list of coordinate for the skyline
    if l == h then                ▷ what method returns once there is no more options to split
        res ← list;
        res.add(arr[0],arr[1]);
        res.add(arr[2],0);
        return res;
    end if
    mid ← (l + h)/2                ▷ Splitting down the middle like merge sort
    listLeft ← calculateSkyline(arr, l, mid);
    listRight ← calculateSkyline(arr, mid + 1, h);
    toReturn ← mergeSkylines(listLeft, listRight)
    return toReturn;
end procedure
procedure mergeSkylines(left, right)
    in: lists of coordinate needed to be merged as left, right
    out: merged and 'sorted' lists of the resulting skyline
    toReturn ← list
    i, j ← 0
    heightLeft, heightRight ← 0
    while i < left.size()&&j < right.size() do
        if left[i][0] < right[j][0] then
            x ← left[i][0];
            heightLeft ← left[i][2];
            maxHeight ← max(heightLeft, heightRight);    ▷ max gets the maximum value of values
entered since only the tallest building can be seen
            toReturn.add((x, maxHeight));
            i ← i + 1;
        else
            x ← right[i][0];
            heightRight ← right[i][2];
            maxHeight ← max(heightLeft, heightRight);
            toReturn.add((x, maxHeight));
            j ← j + 1;
        end if
    end while
    while i < left.size() do    ▷ If one list is bigger than the other we need to add all the rest of the
skylines because we can see all of them.
        toReturn ← left[i];
        i ← i + 1;
    end while
    while j < right.size() do
        toReturn ← right[i];
        j ← j + 1;
    end while
    return toReturn;
end procedure
```

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-8

Collaborators:

The `rand()` function in the standard C library returns a uniformly random number in `[0,RANDMAX-1]`. Does `rand()` $\mod n$ generate a number uniformly distributed in $[0, n-1]$?

Note I: This is the second variant in EPI 5.12.

Note II: When asked questions of this form, you are expected to justify your answer.

Peter Gifford, Ren Wall, Kyle Brekke, Madison Hanson Group: 7 due: 20 September 2019

CSCI 432 Problem 2-9

Collaborators:

Algorithms where we use randomization to find a deterministic answer are known as Las Vegas algorithms. Monte Carlo algorithms also use randomization, but might not always give the right answer; however, they either have a high probability of being correct or close to correct.

(a) Give a Monte Carol algorithm to estimate π.

(b) Let n be the number of random numbers used by your algorithm. Explain why as $n \to \infty$, the expectation of the output for your algorithm is π.

(c) Implement this algorithm and plot a line graph of the values returned for at least 10 values of n.

Note: We can use the function `randReal`(a, b) that returns a random real number between a and b inclusive.