MA/CSSE 473 Day 21

Subset Generation

More Decrease and Conquer Algorithms

MA/CSSE 473 Day 21

- HW 8 is due Friday.
- HW 9 is due Tuesday. Its "late day" period will extend until Friday at noon.
- Don't forget the Convex Hull implementation problem. It is due 24 days after I assigned it (October 21) , and 10 of those days have already passed.
- HW 10 due Tuesday, Oct 19
- HW 11 Due Friday, Oct 22.
- **Student Questions**
- Exam discussion
- Subset Generation
- More Decrease and Conquer algorithms

Exam 1 Discussion

- Exam was a little bit long.
  - No one turned it in more than 2 minutes early
  - My remedy: I used 85 as the max score in the ANGEL gradebook
  - So your score is taken as a percentage of 85.
  - For example, 70 points counts as 82.35%
- Grader has not finished grading HW 7.
- If HW7 grading is finished by tomorrow, midterm grades will be based on HW1-7 + Trominoes, Exam1, and ICQs.

My grade scale approach

- Give many challenging and interesting problems, and adjust the grading scale accordingly:

Grading Scale

Label	Minimum Percent
A	86.5
B+	80.5
B	74.5
C+	68.5
C	62.5
D+	56.5
D	50.5
F	0

Rose Grading Philosophy

- From http://www.rose-hulman.edu/academicpolicies/#grades :
  - Thorough competence to do excellent work in the field is required for the grades of "B" and "B+" which will not be given for mere compliance with the minimum essential standards of the course.

Exam 1, problem 4a

(20) Suppose we have a 64-bit machine. Then we can multiply two 32-bit unsigned integers without overflow. Now suppose that we want to multiply 512-bit positive integers, using one of the divide-and-conquer techniques from class. In the in-class algorithms, the recursion stops when the number of bits gets down to one. In this case we can stop the recursion when the number of bits gets to 32.

a. (5) If we use the standard approach to multiplication, how many 32-bit multiplications are done altogether?

Multiplying two 512-bit numbers requires four multiplications of 256-bit numbers.
Each of those requires four multiplications of 128-bit numbers.
Each of those requires four multiplications of 64-bit numbers.
Each of those requires four multiplications of 32-bit numbers.

Altogether, $4^4 = 256$ multiplications

Exam 1, problem 4b-d

Altogether, $4^4 = 256$ multiplications

b. (5) If we use the algorithm based on Gauss's multiplication formula, how many 32-bit multiplications will be done? **Same idea as above, but only 3 multiplications at each step. $3^4 = 81$**

c. (5) If we only consider the costs of additions and multiplications, and if a 32-bit integer multiplication takes 5 times as long as a 32-bit integer addition is the method from part (b) likely to be faster than the method from part (a)? Show a calculation that leads to your conclusion.
The second algorithm saves 256 – 81 = 165 multiplications.
But it adds 625 – 81 = 544 additions.
If each saved multiplications saves as much time as 5 additions,
the savings is equivalent to 165*5 = 825 additions.
So the answer is "yes".

d. (3) Fill in the blank with the correct number:
If we only consider the costs of additions and multiplications and if a 32-bit integer multiplication takes
__3.3____ times as long as a 32-bit integer addition, then we can expect that the two approaches will take about the same amount of time. Show how you get your answer.
544/165 ≈ 3.3. Either 3 or 4 (or any number in-between) is also an acceptable answer.

Exam 1, problem 5a

Suppose that you use this sub-optimal algorithm: Walk 1 step in one direction, then 4 steps in the other direction, then 9, then 16, then 25, 36, … So at each stage, you walk k^2 steps for some k, then turn around and walk $(k+1)^2$ steps in the other direction if you have not found the door.

(a) (5) After the stage in which you walk k^2 steps in one direction, how far are you from the original starting point? (for example, when k=3, you are 6 steps away from the start; what is the general formula?)

k	distance from start	
1	1	
2	4 – 1 = 3	[note that 3 = 1 + 2]
3	9 – 3 = 6	[note that 6 = 1 + 2 + 3]
4	16 – 6 = 10	[note that 10 = 1 + 2 + 3 + 4]
5	25 – 10 = 15	[note that 15 = 1 + 2 + 3 + 4 + 5]
6	36 – 15 = 21	[note that 21 = 1 + 2 + 3 + 4 + 5 + 6]

So in general, the distance is sum (i, i=1.. k) = k(k+1)/2 . (by formula 2, p 470)

Exam 1, problem 5b

So in general, the distance is sum (i, i=1.. k) = k(k+1)/2 . (by formula 2, p 470)

(10)As a function of N (the distance from the original starting point to the door), what is the worst-case total number of steps taken before finding the door? Give a big-theta estimate and show how you get it.

The worst case is when N is one step past where we end up after the k^{th} stage, for example, at distance $N = k(k+1)/2 + 1$ from the start (for some k). Then in the $k+1^{st}$ stage, we go $(k+1)^2$ steps in the other direction, and in the $k+2^{nd}$ stage, we have to go $(k+1)^2 + 1$ additional steps before we reach the door.

The total number of steps is S = sum (i^2, i=1..k) + 2 $(k+1)^2$ + 1. According to formula 3 on p470, the summation is approximately $k^3/3$, which is $\Theta(k^3)$. The additional two terms are lower order, so S is $\Theta(k^3)$.

From our choice of N, N is $\Theta(k^2)$, so k is $\Theta(N^{1/2})$. Putting these together, S is $\Theta(k^3)$, which is $\Theta(N^{1/2})^3$.
Thus S is $\Theta(N^{3/2})$.

Exam 1, problem 6

(10) A *binary tree* is defined to be either (a) empty, or (b) a root node and two subtrees, T_L and T_R, each of which is a binary tree. The *height* of a binary tree is the length of the longest path from the root node to another node in the tree. Note that a tree with one node has height 0, and the empty tree has height -1. We can use the notation N(T) = the number of nodes in tree T, and H(T) = the height of tree T. Use mathematical induction and the given definition of binary tree to carefully prove that for every binary tree, $N(T) \le 2^{H(T)+1} - 1$. Be sure that your description makes it clear where and how you apply the induction hypothesis.

The induction is on H(T). Base case: $H(T) = -1$. $N(T) = 0$. $2^{H(T)+1} = 1$. So $N(T) \le 2^{H(T)+1} - 1$.

Induction step. Consider a nonempty tree T whose height is H(T). Assume by induction that the property is true for all binary trees whose height is smaller. In particular, T's subtrees are shorter, so the property holds for each of them. We also know that $H(T_L) \le H(T) - 1$, and $H(T_L) \le H(T) - 1$. Putting all of this together:

$$N(T) = 1 + N(T_L) + N(T_R) \le 1 + (2^{H(T_L)+1} - 1) + (2^{H(T_L)+1} - 1)$$ by the induction hypothesis

$$\le 1 + (2^{H(T)+1-1} - 1) + (2^{H(T)+1-1} - 1)$$ by the inequality from the above paragraph

$$= 2^{H(T)} + 2^{H(T)} - 1 = 2(2^{H(T)}) - 1 = 2^{H(T)+1} - 1$$ by simple algebra

Exam 1, problem 8

(5) If we use brute force integer multiplication and addition, and brute-force matrix multiplication, what is the big-theta running time (in terms of n) for multiplying n n×n matrices of n-bit integers? Circle one

$\Theta(n^2)$ $\Theta(n^3)$ $\Theta(n^4)$ $\Theta(n^5)$ **$\Theta(n^6)$** $\Theta(n^7)$ $\Theta(n^8)$ $\Theta(n^9)$ $\Theta(n^{10})$ $\Theta(n^{11})$

Each numeric multiplication is n^2. A single matrix multiplication requires n^3 of those, and there are n-1 matrix multiplications.

Partial credit: : 2 points if you said n5 or n7.

Interesting note: This is the only problem that I re-used, unchanged, from the Fall 2008 Exam 1 that I emailed to everyone out on Monday.

Surprisingly, of all the problems on the exam, the one you got to see beforehand is the one whose overall percentage score was lowest.

Questions on any other problems?

Subsets of a set

BACK TO GENERATION OF COMBINATORIAL OBJECTS

Generate All Subsets of a Set

- Sample Application:
  - Solve the knapsack problem
  - In the brute force approach, we try all subsets
- If A is a set, the set of all subsets is called the **power set** of A, and often denoted 2^A
- If A is finite, then $\left|2^A\right| = 2^{|A|}$
- Thus we know how many subsets we need to generate.

Generate all Subsets of $\{a_0, \ldots, a_{n-1}\}$

- Decrease by one:
- Generate S_{n-1}, the collection of the 2^{n-1} subsets of $\{a_0, \ldots, a_{n-2}\}$
- Then $S_n = S_{n-1} \cup \{ s \cup \{a_{n-1}\} : s \in S_{n-1}\}$
- You'll do details of this in the homework.
- Another approach:
  - Each subset of $\{a_0, \ldots, a_{n-1}\}$ corresponds to a bit string of length n, where the i^{th} bit is 1 iff a_i is in the subset

Another approach:

- Each subset of $\{a_0, \ldots, a_{n-1}\}$ corresponds to an bit string of length n, where the i^{th} bit is 1 if and only if a_i is in the subset

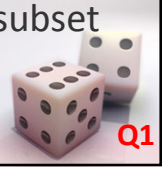
```
def allSubsets(s):
  n = len(s)
  subsets=[]
  for i in range(2**n):
    subset = []
    current = i
    for j in range (n):
      if current % 2 == 1:
        subset += [s[j]]
      current /= 2
    subsets += [subset]
  return subsets
```

Output:

```
[[], [0],[1],
[0, 1], [2],
[0, 2],
[1, 2],
[0, 1, 2]]
```

Gray Code

- Named for Frank Gray
- An ordering of the 2^n n-bit binary codes such that any two consecutive codes differ in only one bit
- Example:
  000, 010, 011, 001, 101, 111, 110, 100
- Note also that only one bit changes between the last code and the first code.
- A Gray code can be represented by its **transition sequence:** which bit changes each time
  In above example: 1, 0, 1, 2, 1, 0, 1
- In terms of subsets, the transition sequence tells which element to add or remove from one subset to get the next subset



Q1

Generating a Transition Sequence

- Binary Reflected (Gray) Code
- $T_1 = 0$
- $T_{n+1} = T_n , n, T_n^{reversed}$
- What are T_2, T_3, T_4
- Show that Tn is always a palindrome and that $T_{n+1} = T_n , n, T_n$

Q2 should say, "Show the transition sequence for …"

Q2-3