**Homework (Tell me if I forgot something) CISC 3220 February 21, 2016**

On exam, some % if you write nothing for a question. 0% for garbage.

The homework questions prepare you for the exam, but will not necessarily be repeated exactly.

You may assume that the size of the input is odd or even for your convenience. Please state if you make this assumption.

**Chap1-2**

**problem**

**algorithm**

**Analysis** examines:

* Correctness
  1. Loop invariant
* Time & Space Efficiency
  1. Empirical
  2. Theoretical – Worst, average, best
* Optimality

**search and sort examples**

**prefix-sum**

* 2.1-1,2,4
* 2.2-2, 3
* 2.3-1 to 2.3-6
* Think about 2-1 and 2-2

**Chap 3**

**growth**

**Asymptotic** analysis

**O, Ω, Θ, (o, ω), O(1)**

* *f* **∈***O(g)* **⇔***g* **∈****Ω***(f)*
* *O(f + g) = O(max(f, g))*
* 3.1-1, 3, 4, 5, 8
* 3-1 Know, don’t prove
* 3-3 b is a challenge



**Three proofs of above**

* **Prove lg(*n*!) <= *n*lg*n***

**Two very similar direct proofs from definition**

**One different proof**

**Chap 4**

**Reurrence – solve (get closed form or bound)**

* Guess and prove
  + T(n)=2T(n/2)+n, T(1)=1
    - Guess: T(n)=nlgn+n
    - Prove by induction
  + T(n)=3T(n-1) + 2, T(1)=2
    - Expand it and write a series and solve. (hint: geometric series)
    - Can you guess by calculating a few values?
    - Prove by induction
* Draw **tree** - - use to solve
  + 4.4-1(4.2-1 in 2nd) ignore floor
    - Answer is *3nlg3-2n ~ 3n1.5-2n = Θ(nlg3)*
  + 4.4-6 (4.2-2 in 2nd)
  + T(n)=2T(n/2)+n2 
    - Will get Geometric series.
    - T(n) =2n2-n
    - Ignore base constant so really *Θ(*n2)
* **Master Theorem**
  + 4.5-1,3 (4.3-1,3 in second ed’n)
  + Can you do 4.5-2 (4.3-2 in 2nd)?
  + 4-1 a-f

**Chap 6**

**Storing a binary tree in an array**

**Heap**

**Heapify**

**Build-Heap**

**Heapsort**

**Implementing a priority queue as a heap**

**Extract-Max**

* 6.1-1 to 6.1-6
* 6.2-1 and 6.3-1
  + you don’t have to redraw each time- can show work as cross out and rewrite
  + give the final array
* 6.2-3, 6.2-4
* 6.3-1, 6.3-2
* 6.4-1 ,
* 6.5-1
* Problem 6-3

**Chap 7**

**QuickSort**

**Hoare –Partition**

**Cogan-Partition**

**Randomized Partition (Sherwood Algorithm)**

* Problem 7-1 a
* 7.1-2 ? Does second half make sense for Hoare-Partition?
* 7.1-3 to 7.1-4
* Does Cogan-Partition work?
* 7.2-1 to 7.2-4
* For small arrays, InsertionSort is faster than QuickSort, so call InsertionSort when you get down to a certain size.

Or Read 7.4-5

**Chap 8**

**decision trees**

**optimal sort**

**CountingSort**

**RadixSort**

* 8.1-1, 8.1-2
* 8.2-1 to 8.2-4
* 8.3-1to 8.3-2

**Chap 9**

**max, min, max and min in 1.5n time**

**randomized select**

* 9.2-4 also trace it without randomizing
* 9.3-7 Don’t worry about endpoints, parity (odd or even), floors and ceilings.
  + Optional: worry about them ☺
* Maximum Difference
* Input: A[1..n]
* Output: Maximum difference between two elements
* Give an algorithm that takes \_\_ comparisons
  + n2
  + n(n+1)/2
  + nlgn
  + 2n
  + 1.5n
* 9-1 May need Cogan-Partition for c.

**Chap 15**

**Dynamic Programming**

* optimal substructure
* overlapping subproblems
* recursive solution
* bottom-up solution

**Problems**

* Fibonacci
* binomial coefficients
* matrix chain multiplication
* longest common subsequence
* edit distance variation of 15-5
* 0-1 knapsack
* prefix palindrome
* 15.2-1          i.e. Trace  Print-Optimal-Parens(s, i, j)
* 15.2-2
* 15.4-1  Not by sight.  Use the algorithms.
* 16.2-2
* Simplification of Problem 15-2:
  + Develop a linear algorithm (one dimensional array) to find the  longest palindrome prefix (rather than substring) of a given string. Analyze it.
  + E.g. “nursesrunaway” has palindrome prefix “nursesrun”
* Be able to give the formulas and fill the tables for all examples and homework. I doubt I will give you an original on an exam.

**Chap 16**

**Greedy Algorithms**

* optimal substructure
* greedy choice property

**Problems**

* activity selection
* fractional knapsack
* Huffman coding
* coin changing
* 16.1-3
* C={a,b,c,d,e}     f(C)={50, 25, 15, 41, 75}  Note:  I changed 40 to 41.  What are all the codewords? What does 00010001 represent?
* 16.3-3
* 16-1 a, c

**Chap 22**

**Graph, subgraph**

**Path, Connected**

**Cycle, acyclic, tree, forest**

**directed, strongly connected**

**complete**

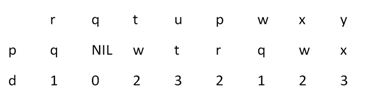
**adjacency list, adjacency matrix**

**transpose**

**weighted**

**Breadth-first-search (BFS), Depth-first-search (DFS)**

* 22.2 – 1(directed), 2(undirected)
* Trace/Execute Print-Path( G, q, y) for the following:



* 22.3-2 Start at r. What would happen if you started at q? You do not need to classify the edges.
* 22.3-3
* B-1a

**Chap 23**

**MST- Kruskal , Prim**

* Know how to give me the order of edges processed for algorithms of Kruskal and Prim. (pushed/popped and extracted)

**Chap 24**

**Dijkstra’s single source shortest paths tree**

* 24.3-1

Use either style of Figure 24.6 or my style

* 24.3-2

**Chap 32**

**String Matching**

**Finite Automata**

**Knuth Morris Pratt (KMP)**

* 32.4-1
* For a given Pattern:
  + figure out π
  + draw a KMP FA
  + Given a text string, trace the states through which the KMP goes to match the pattern (once).

**Chap 34**

1. Input: A **sorted** sequence of numbers <a1,....,a n> and a number B.

Output: True if there exist ai1 and ai 2 , such that B = ai 1 + ai 2. False otherwise.

Find a **linear** time algorithm to solve this problem and analyze its complexity.

**2. Subset Sum** ( Section 34.5.5 of text)

Input : A set S ⊆ **N**(Integers) with |S|=k, and a target t ∈ **N**.

Output: True if there exists a subset S'⊆S such that t = Σ s. s∈S'

False otherwise.

In other words: “An instance of the subset-sum problem is a pair (S, t), where S is a set of positive integers and t is a positive integer. This decision problem asks whether there exists a subset of S that adds up exactly to the target value t.”

1. Can you find a O(n), O(n2), O(nk) algorithm? Hint: It’s Θ(2k).
2. Explain why?
3. If T(n)=2n microseconds . How many hours will it take to process a set of size 32?

How many years will it take to process a set of size 52?

1. If you are given an instance of the Subset Sum problem and a proposed solution, how long will it take you to ***verify*** that the solution is correct?

3. **Set Partition** (exercise 34.5-5)

Input : A set S ⊆ **N**(Integers.

Output: True if there exists subset S1,S2 ⊆ S such that Σ s = Σ s, S1∩S2=Ø and S1US2=S.

s∈S1 s∈S2

False otherwise.

In other words: “Given a set S of integers, is there a way to partition (divide into mutually exclusive and exhaustive subsets) S into S1 and S2 such that the sum of the numbers in S1 equals the sum of the numbers in S2?”

If you had an “Oracle” (source of wisdom) to solve the Subset Sum problem, how much more time would you need to solve Set Partition? Hint: Find the sum of the elements. Present the Oracle with an instance of the Subset Sum problem. Use its answer to solve your Set Partition Problem efficiently.

4. Please note that both the above questions (2 and 3) could have been asked about a multiset. (What’s that?)

**Decision Problem, Optimal Value, Optimization Problem**

**P**

**NP**

**NP-hard**

**NP-Complete (NPC)**

**≤P, Polynomial Reductions**

**NPC Problems:**

* cSAT
* Satisfiability
* Subset Sum (Section 34.5.5)
* Set Partition
* Clique Problem
* Vertex Cover
* Independent Set
* TSP
* Hamiltonian Cycle
* 3-colorable
* When you prove a problem is NP-complete:
  + What assumption are you making?
  + What are the two things you must prove?

**Chap 35**

**Approximation Algorithms**

TSP

Greedy 0-1 knapsack (Greedy fractional knapsack gives exact answer).

**Not in text**

Know about Erdos number (Kevin Bacon Game).