**Regis University CC&IS**

**CS324 Algorithms & Analysis**

**Alternate Homework 3 v18**

**Assignment**Place your answers directly under each question below -- do not delete the questions.

***Divide-and-Conquer***

1. Characterize each of the following recurrence equations using the master theorem  
   (assuming that: Textbook: **T(n) = c** for n < d, and **T(n) = aT(n/b) + f(n)** for n >= d

Content: **T(n) = c** for n=1, and **T(n) = aT(n/b) + f(n)** for n > 1)

**Explain** which case was used, and why.

And provide the final results. (6 *pts each for 18 pts total*)

1. *T*(*n*) = 27*T*(*n/*3) + *n*2
2. *T*(*n*) = 8*T*(*n/*2) + (*n* log *n*)3
3. *T*(*n*) = 10*T*(*n/*2) + *n*3
4. Use Stassen’s matrix multiplication algorithm to multiply the following matrices:  
   Show **all** steps (*10 pts*).

A = B =

1. Give an example of a set of at least 5 points, where only one of the points is a maxima point.  
   Be sure to specify which point is the maxima point (*5 pts*).
2. Use an algorithm to find the maxima set from the following set of 12 points:

{ (3,2), (1,1), (5,3), (7,5), (4,5), (6,7), (2,6), (1,8), (9,4), (8,3), (6,2), (4,1),) }

**Explain** how the steps you followed to get the answer (*12 pts*).

***Dynamic Programming***

1. Let S = {A, B, C, D, E, F, G} be a collection of objects with benefit-weight values of

A:(8, 2), B:(5, 3), C:(8, 3), D:(7, 4), E:(14, 5), F:(10, 6), G:(11, 6)

Note that the objects are already ordered by ascending weights.

Use dynamic programming to find an optimal solution to the 0-1 knapsack problem,

if the sack that can hold objects with total weight 12.

Show the dynamic programming benefits array (Bi,j) that would be generated.

Then show the path followed back through your table to determine the solution.  
Finally, give the maximum benefit achieved and which objects would be included. (*20 pts*)

1. Given a set of 7 job requests, (si, fi, bi), defining the start times, finish times, and benefits of each job:

1:(2, 3, 5), 2:(1, 4, 8), 3:(3, 4, 4), 4:(3, 5, 7), 5:(4, 6, 5), 6:(6, 7, 3), 7:(5, 9, 9)

Note that the jobs area already ordered by end time (and by start time within same end time).

Using dynamic programming, solve the job scheduling problem for this set of job requests.

Show the predecessor array and the benefits array.

Then give the jobs that would be included in the optimal solution, and the total benefit. (*15 pts*)

1. Show the longest common subsequence table, L, for the following two strings:

"DENVERCOLORADO"

"EVERYMONDAY"

Use dynamic programming to find a longest common subsequence between these strings. (*20 pts*)

Show your work as a filled array similar to the exercises in Content 9.5.2, with the **first** string as the **rows**, and the **second** string as the **columns** of your dynamic programming array.

After creating the array, show the path taken back through the array to determine the longest subsequence, and give the subsequence found.

**Submission**

This homework assignment is due by midnight of the date listed on the **Course Assignments by Week** page of the Content.

* Before you submit your homework, append your last name to the front of the Word doc filename. For examples: **Smith-CS324-AltHwk3-v18.docx**
* Submit your **.docx** file to the **Hwk Assn 3** Submission Folder (located under **Assignments** tab in online course.

***WARNING:****Homework submitted more than* ***3 days*** *past the due date will* ***not*** *be accepted,  
and will receive a grade of 0.*