# Data Structures and Algorithms

# INFO 6205

# Homework 9

# NOTE: This homework is not collected for grading but it will be on the Final-exam

1. Explain:

Cell, Gene, Chromosomes, DNA

Human Genome Project?

Cell Nucleus

Genomic Language

DNA Mutations, name 3 mutations and explain

2. Consider the article on Genetic Algorithm (GA)

a) The following model describes the biology in human. Describe the Model,

How does it work?

b) What is the goal of Genetic Algorithm? Explain with Example

c) Genetic Algorithm mimics its model from science of biology (True/False)? Explain

3. Genetic Algorithm:

` a) What are the steps of Genetic Algorithm?

b) Read the example Java code (Lecture10Notes) as how it relates to steps in (a)

c) Compile and run the code, explain the results.

d) What are possible termination points?

e) Change the code to consider for 8 genes. Compile and Run.

f) Compare (c) and (e), explain results

4. Consider the following Text and Pattern

Text: ABCADBABCBABABCDABCDABDE

Pattern: DAB

a) Apply Brute-Force substring search algorithm to scan Pattern in

Text string. Show step-by-step of the algorithm. What is time complexity?

b) Apply Robin-Karp substring search algorithm to scan pattern in the

text string. Show step-by-step of algorithm. Write Java code

for the algorithm for input data. What is time complexity?

c) What is the difference between the two time complexity?

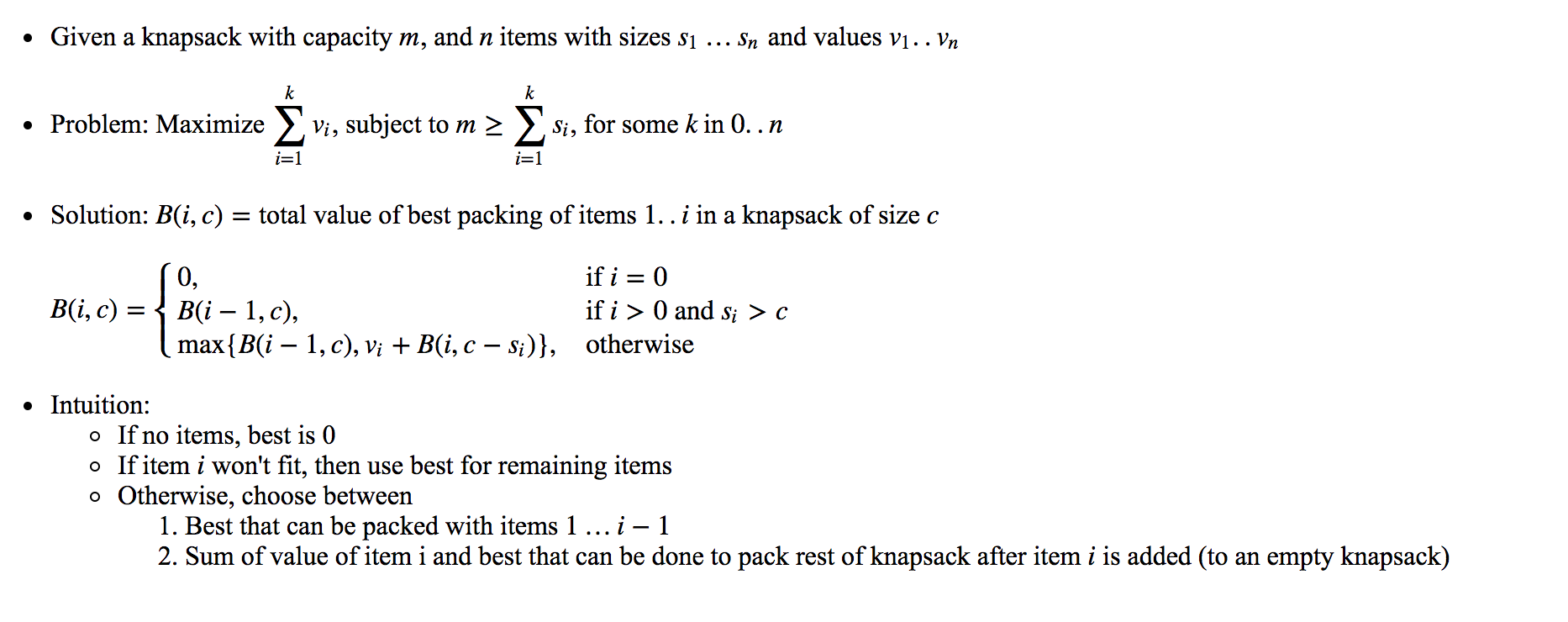
5. Consider Knapsack problem: There are six items with the following:

Name A B C D E F

Size 4 6 7 3 2 5

Value 3 5 9 6 4 8

A) What are the possible solutions with capacity m=24?

B) Explain the following Knapsack Optimization Model

6. Consider paper “Genetic Algorithms for Balanced Minimum Spanning Tree Problem”

a) Read Abstract —what does the paper intends to accomplish?

b) Develop Graph(s) that formulates ABSTRACT

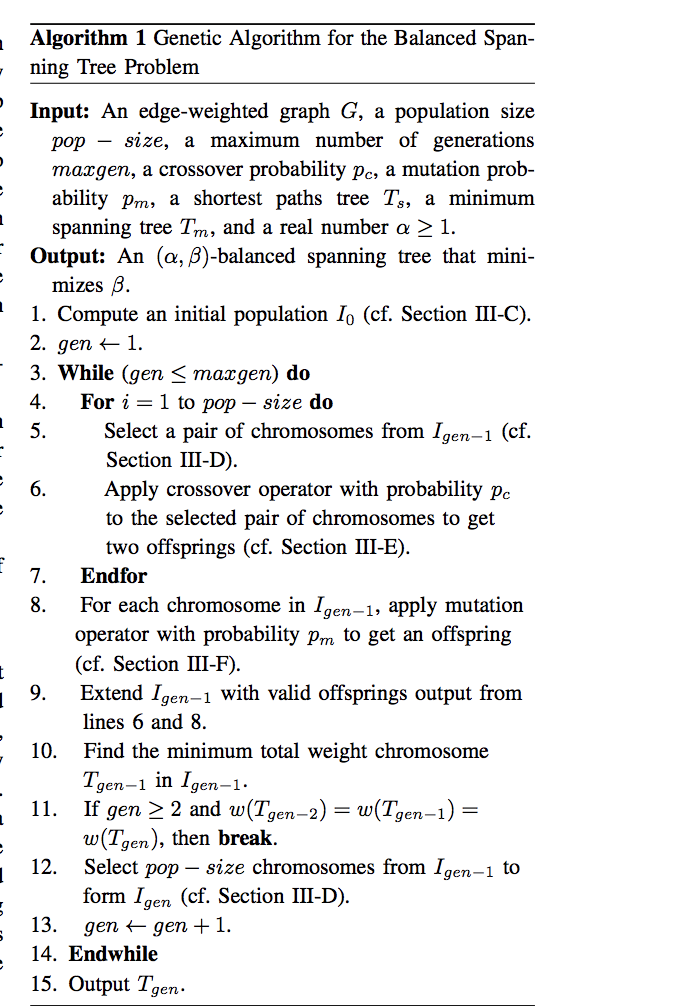
c) What elements in Algorithm-1 relates to Genetic algorithm, explain

d) Formulate a Model data structures to initialize population for genetic algorithm

e) Explain how Algorithm works to output Balanced Minimum Spanning Tree

Reference: Balanced Spanning Tree

<https://stackoverflow.com/questions/4795983/balanced-spanning-tree-t-from-undirected-graph>



7. Consider this article, it discusses two approaches, Recursive and Dynamic programming.

<https://www.techiedelight.com/longest-common-subsequence/>

a) Describe this recursive Tree structure, There is an important problem with this structure.

What is it? Discuss

b) Compare recursive and dynamic programming approaches, what benefits does

dynamic programming provide?

8. Consider the following two string sequences, find Longest Common subsequences

s1=GACGGGCATTAG

s2=GATCGGAATAAG

9. Consider Sequences x = GACGGAGATTAG and y =  GATCGCGAATAG. Assume that

the match score is +1, and the mismatch is -1, and gap penalties is -1.

A) Fill out the dynamic programming table for a global alignment between x and y.

B) Draw arrows in the cells to store traceback information.

C) What is the Score of the optimal global alignment and what Alignment(s)

achieves this score?

10. Answer these questions:

What is Regular Expression?

What is NFA + DFA= Finite State Machine (FSM)

What is Finite?

What is the goal of Finite State Machine?

What is the difference between DFA and NFA?

Are we interested in Deterministic or Non-deterministic FSM? Why?

11. For the following Regular Expression (RE) Input Strings

NOTE: Study Rules on Regular Expression Slides I provided

a) Convert each RE to DFA

ab\*a abc\*|x+y w(x|y)\*z a(xy)\*

b) Convert RE to NFA

a\*bc a\*|mn (w|x)\*

c) Convert X(X|Y)\*|Z to NFA and then to DFA

12. Consider regular expression **(a|b)\*abb** with the following FSM code.

A) What is this program is used for?

B) If you have a new expression X(X|Y)\*|Z+X, what would you change

the code and how? Compile and run the code, discuss the results.

#include <iostream>

#include <cassert>

#include <string>

using namespace std;

typedef int fsm\_state;

typedef char fsm\_input;

bool is\_final\_state(fsm\_state state)

{

return (state == 3) ? true : false;

}

fsm\_state get\_start\_state(void) { return 0; }

fsm\_state move(fsm\_state state, fsm\_input input)

{

// our alphabet includes only 'a' and 'b'

if (input != 'a' && input != 'b') assert(0);

switch (state)

{

case 0:

if (input == 'a') { return 1; }

else if (input == 'b') { return 0; }

break;

case 1:

if (input == 'a') { return 1; }

else if (input == 'b') { return 2; }

break;

case 2:

if (input == 'a') { return 1; }

else if (input == 'b') { return 3; }

break;

case 3:

if (input == 'a') { return 1; }

else if (input == 'b') { return 0; }

break;

default:

assert(0);

}

}

bool recognize(string str) {

if (str == "") return false;

fsm\_state state = get\_start\_state();

string::const\_iterator i = str.begin();

fsm\_input input = \*i;

while (i != str.end()) {

state = move(state, \*i);

++i;

}

if (is\_final\_state(state))

return true;

else

return false;

}

// simple driver for testing

int main(int argc, char\*\* argv) {

recognize(argv[1]) ? cout < 1 : cout < 0;

return 0;

}