Module 11 Problems

1. [12 pts] Given input alphabet {A, B, C, D}, construct the DFA for the string

A B C D A B D

NOTE: The spaces in the above string and aphabet are for readability.
The spaces are **not** part of the pattern or alphabet.

Show a trace of the KMP algorithm and your DFA on the target string

A B C A B C D A B C D A B D

NOTE: Again, the spaces are for readability.

2. [12 pts] Regular expressions (or regexs) are often extended with convenience operations.
Assuming the alphabet is the lower case letters a,b,c,d,e; write the regular expression for the following convenience operations:

2.1 [3pts] Wildcard (usually the '.' character): will match any character in the alphabet

2.2 [3pts] Kleene +: Like *, but the regular expression must have at least 1 instance of the root regex

Example: (ac)+ matches "ac", "acac"
(ac)+ does **not** match "" (the empty string)

2.3 [3pts] bounded closure: Match a finite set of concatenations.

Example: (ab){3,5} matches "ababab" or "abababababab"
(ab){3,5} does **not** match "ab" or "abab" or "abababababab"

2.4 [3pts] character range:
Example: a[b-d] matches ab, ac, or ad

3. [18 pts] Given the regex over the alphabet {A,B,C,D,E,F,G}

((A | B)* | C D* | E F G)*

NOTE: The notation for the regex above uses spaces for readability.
The space character is **not** part of the pattern

3.1 [9 pts] Construct the NFA state machine and digraph of e-tranisitons for the given regex

NOTE: The above regex *may not be* fully parenthesized.
If needed, you may add enough parentheses make the regex fully parenthesized.

3.2 [9 pts] Show the full set of state transitions when your NFA recognizer is applied to

A B B A C E F G E F G C A A B

NOTE: The spaces in the above string are for readability.
The space character is **not** part of the source string.

4. [14 pts] A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S.
For instance, if S is

5, 15, −30, 10, −5, 40, 10

then

15, −30, 10

is a contiguous subsequence.

The subsequence

5, 15, 40

is **not** contiguous.

Give a linear-time algorithm for the following task:

Input: A list of numbers, a1,a2,...,an
Output: A contiguous subsequence of maximum sum.
NOTE: A subsequence of length 0- has sum 0.

For the preceding example sequence,the answer would be

10, −5, 40, 10

with a sum of 55.

Hint: For each $j \in \{1, 2, \ldots, n\}$, consider contiguous subsequences ending exactly at position j.

5. [14 pts] You are given a string of n characters s[1 . . . n], which you believe to be a corrupted text document in which all punctuation and whitespace has vanished. A sample input is:

itwasthebestoftimesitwastheworst

You wish to reconstruct the document using a dictionary. The dictionary is available in the form of a Boolean function

`dict(w)`:
for any string w, `dict(w)` = true if w is a valid word in the dictionary
`dict(w)` returns false otherwise.

5.1 [9 pts] Give a dynamic programming algorithm that determines whether the
string s can be reconstituted as a string of valid words.
The running tine should be no worse that O(|s|2), assuming a call to the dict function takes O(1).

5.2 [5 pts] In the event that the string is valid, make your algorithm output the corresponding sequence of words.

6. [16 pts] Best Matrix Multiply Order:
Let

A1A2A3A4...An

be a sequence of matrices that must be multiplied together.
Matrix Multiply is associative, so you may parenthesize the multiplication in whatever way works best.
Some reminders: An m x p matrix can be multiplied by a p x q matrix giving an m x q matrix. The cost of multiplying the 2 matrices is m * p * q.

Example: Given the sequence

A1A2A3
where A1=10×10, A2=10×10, A3=10×1.
Choosing (A1 * A2) * A3 gives:
cost of A1×A2=10×10×10=1000
cost of(A1×A2)×A3=10×10×1
total cost = (1000) + 100 = 1100

On the other hand, choosing A1×(A2×A3)
cost of A2×A3=10×10×1=100
cost of A1×(A2×A3)=10×10×1=100
total cost = 100 + 100 = 200

So, 2nd choice is clearly best.

Your problem: Given a sequence of matrices and their dimensions, construct an algorithm that gives the order of matrix multiplies
that yields the lowest number of operations.

7. [14 pts] Minimum Edit Distance:
You have at your disposal 3 character editing operations:

1. d: delete a char
2. i: insert a char
3. c: change 1 char into another char

You are given 2 strings s,t.
Your problem is to always find the minimum # of operations to turn string s into string t.

Example: s = 'cast', t='cats'

Option 1: c s->t [pos 3]; c t->s [pos 4] has 2 operations
Option 2: d a [pos 2]; c s a [pos 2];i s at the end. This has 3 operations.

Option 1 is clearly better option 2.