you @ cs. ualberta. ca

Exercise Set #6

CMPUT 204
Department of Computing Science
University of Alberta

Problem 1. In general, the hardest task of dynamic programming is to find and define the "right" recursion for the problem — a recursion that makes exponentially-many recursive calls in a direct implementation, and yet all of the calls are in some poly-size domain. For each problem below, find such a recursion. Then explain how you solve the problem in a bottom-up fashion by using a suitably defined array/table and a formula to fill the array based on previously-filled cells. For this problem you do not need to write a pseudocode as part of your explanation.

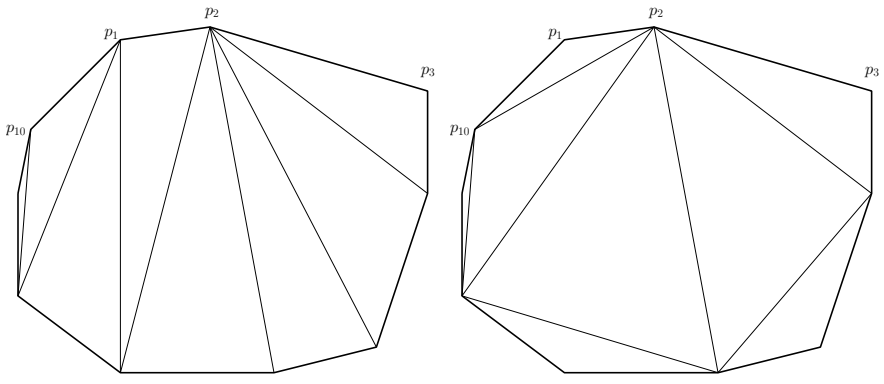
a. The LONGEST SUB-PALINDROME problem: given a sequence $X = \langle x_1, ..., x_n \rangle$ find the longest subsequence $\langle x_{i_1}, ..., x_{i_k} \rangle$ such that $i_j < i_{j+1}$ for any j, and that is a palindrome: k is even and the inverse sequence $\langle x_{i_k}, x_{i_{k-1}}, ..., x_{i_1} \rangle$ is identical to the original sequence $\langle x_{i_1}, ..., x_{i_k} \rangle$. (E.g., "abba" is a sub-palidrome of "tablebrand.")

b. The LONGEST SIMPLE PATH problem: Suppose that we are given a directed acyclic graph $G = (V, E)$ with real-valued weights and two distinguished vertices s and t. The goal is to find a longest weighted simple path from s to t.

Problem 2.

There are n Hudson's Bay Company posts on the River Koksoak. At any of these posts you can rent a canoe to be returned at any other post downstream (it is next to impossible to paddle against the current.) For each possible departure point i and each possible arrival point j the company's tariff gives the cost of a rental between i and j which is $C[i, j]$. However, it can happen that the cost of renting from i to j is higher than the total cost of a series of shorter rentals, in which case you can return the first canoe at some post k between i and j and continue the journey in a second canoe. There is no extra charge for canoe change. Give an efficient algorithm to determine, given n and costs $C[i, j]$, the sequence of canoe rentals with minimum cost for a trip from post 1 to post n. What is the running time of your algorithm?

Problem 3. Suppose we are given a convex n-sided polygon P (a polygon is called convex if the angle of each interior corner is less than 180 degrees). A *triangulation* of P is adding diagonals connecting the vertices of P so that no two diagonals intersect each other (except at a corner if they share that corner) and every interior face (region) is a triangle. The following pictures show two different triangulations of a convex polygon (with $n = 10$).



The cost of a triangulation is the sum of the lengths of the diagonals added. For any two corners p_i and p_j of P (with coordinates x_i, y_i and x_j, y_j) assume we have a method that can compute the length of the diagonal between p_i and p_j in constant time and also compare these lengths in constant time. Present an efficient algorithm for computing a minimum cost triangulation of P. Explain why the algorithm is correct and do the analysis of the running time as well.

Problem 4. Suppose we are given an n-node rooted tree T, such that each node v in T is given a weight $w(v) \in \mathbb{R}^+$. An independent set of T is a subset S of the nodes of T such that no node in S is a child or parent of any other node in S. Design an efficient dynamic programming algorithm to find the maximum-weight independent set of the nodes in T, where the weight of a set of nodes is simply the sum of the weights of the nodes in that set. What is the running time of your algorithm?

Problem 5. A company named $RT\&T$ has a network of n switching stations connected by m high-speed communication links. Each customer's phone is directly connected to one station in his or her area. The enginners of $RT\&T$ have developed a prototype video-phone system that allows two customers to see each other during a phone call. In order to have acceptable image quality, however, the number of links used to transmit video signals between the two parties cannot exceed 4. Suppose that $RT\&T$'s network is represented by a graph. Design an efficient algorithm that given this graph, computes for each station, the set of stations it can reach using no more than 4 links.

Problem 6. The square of an undirected graph $G(V, E)$ is a new graph $G^2(V, E')$ where edge $uv \in E'$ if and only if the distance between u and v in G is at most 2, i.e. either $uv \in E$ or there is a $w \in V$ such that $uw \in E$ and $wv \in E$. Describe efficient algorithms for computing G^2 for each of the following two cases:

i) G is represented in adjacency list,

ii) G is represented in adjacency matrix.

Analyze the running time of your algorithm.

Problem 7. A digraph $G = (V, E)$ is called a *one-way connected graph* if for every two vertices $u, v \in V$, either there is a path from u to v or there is a path from v to u (but never both, so u and v can never be strongly connected).

a. State a necessary and sufficient condition for a digraph to be one-way connected. Prove its correctness (i.e., prove it is a necessary condition and also a sufficient condition).

Note: Here is an incorrect answer. Consider the condition "every pair of vertices u and v are not strongly connected". It is obviously a necessary condition, since a one-way connected graph cannot have both a path from u to v and a path from v to u. While it is NOT a sufficient condition, since a digraph satisfying this condition may not be a one-way connected graph (e.g., in a digraph with three vertices $1, 2, 3$ and two edges $\langle 1, 2\rangle$ and $\langle 3, 2\rangle$, each pair of vertices are not strongly connected, and the digraph is also not one-way connected since there is no path between 1 and 3).

b. Describe an efficient algorithm to determine whether or not an input digraph $G = (V, E)$ is one-way connected. Analyze the running time of your algorithm. (Hint: Make use of the condition you state in part **a**.)

Problem 8.

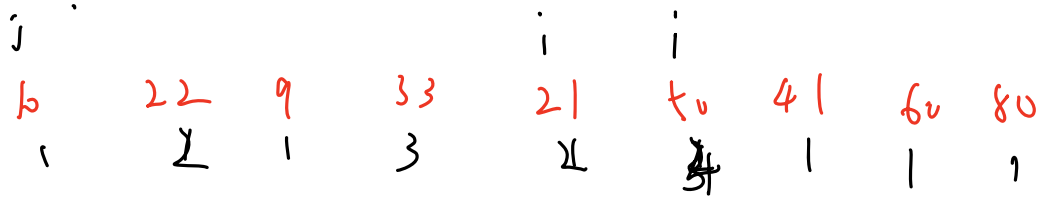
(i) Exercise 22.3-10 of CLRS p612.
(ii) Problem 22-1 (both a and b) of CLRS p621
(iii) Problem 22-2 of CLRS p621. Answer questions **a**, **b**, and **e**.

The following problems are for additional practice. They will not be considered for quizzes.

Problem 9. One can find interesting problems that can be solved by DP from the web. I'd suggest this web site: `https://www.geeksforgeeks.org/top-20-dynamic-programming-interview-questions/` for additional problems, which are claimed to be "top 20 DP Interview Questions." Note that DP is not only for optimization problems (just like Fibonacci number problem, which is not an optimization problem). You may take a look at the following subset:

- Longest Increasing Subsequence
- Edit Distance (a simpler version of Problem 15-5 of CLRS p406-408)
- Minimum Partition
- Subset Sum Problem (Here the given integers are nonnegative. For arbitray integers, the problem is unlikely poly-time solvable, a so-called NP-hard problem)
- Optimal Strategy for a Game (an interesting two-player game problem)
- Word Break Problem (said to be a Google interview question; not an optimization problem)
- Egg Dropping Puzzle (The same problem we have seen in a previous Exercise)

Problem 10. Exercise Exercise 22.5-7 of CLRS p621.



Problem 1. In general, the hardest task of dynamic programming is to find and define the "right" recursion for the problem — a recursion that makes exponentially-many recursive calls in a direct implementation, and yet all of the calls are in some poly-size domain. For each problem below, find such a recursion. Then explain how you solve the problem in a bottom-up fashion by using a suitably defined array/table and a formula to fill the array based on previously-filled cells. For this problem you do not need to write a pseudocode as part of your explanation.

a. The LONGEST SUB-PALINDROME problem: given a sequence $X = \langle x_1, ..., x_n \rangle$ find the longest subsequence $\langle x_{i_1}, ..., x_{i_k} \rangle$ such that $i_j < i_{j+1}$ for any j, and that is a palindrome: k is even and the inverse sequence $\langle x_{i_k}, x_{i_{k-1}}, ..., x_{i_1} \rangle$ is identical to the original sequence $\langle x_{i_1}, ..., x_{i_k} \rangle$. (E.g., "abba" is a sub-palidrome of "tablebrand.")

b. The LONGEST SIMPLE PATH problem: Suppose that we are given a directed acyclic graph $G = (V, E)$ with real-valued weights and two distinguished vertices s and t. The goal is to find a longest weighted simple path from s to t.

a. Omit the first character and recurse on the remaining sequences; Omit the last character and recurse on the remaining sequence.

$$\text{Longest\_Sub\_Palindrome} = \begin{cases} 1 & \text{if } i=j \\ LSP[i+1,j-1] + 2 & \text{if } x[i] = x[j] \\ \max[LSP[i+1..j], LSP[i..j-1]] & \text{if } x[i] \neq x[j] \end{cases}$$

agbdba

	0	1	2	3	4	5
0	1	1	1	1	3	5
1		1	1	1	3	3
2			1	1	1	1
3				1	1	1
4					1	
5						1

b. create a topological order of of the directed graph G. Array dist[V] stores the longest path from s to the vertex V.

recursion: $dist[V] = \max_u \{ dist[u] + w[u,v] \}$, u is any parent of vertex V, and w[u,v] is the edge weight.

Problem 2.

There are n Hudson's Bay Company posts on the River Koksoak. At any of these posts you can rent a canoe to be returned at any other post downstream (it is next to impossible to paddle against the current.) For each possible departure point i and each possible arrival point j the company's tariff gives the cost of a rental between i and j which is $C[i, j]$. However, it can happen that the cost of renting from i to j is higher than the total cost of a series of shorter rentals, in which case you can return the first canoe at some post k between i and j and continue the journey in a second canoe. There is no extra charge for canoe change. Give an efficient algorithm to determine, given n and costs $C[i, j]$, the sequence of canoe rentals with minimum cost for a trip from post 1 to post n. What is the running time of your algorithm?

let $T[1,k]$ be the cheapest cost from post 1 to post k, for $k = 1, \ldots, n$

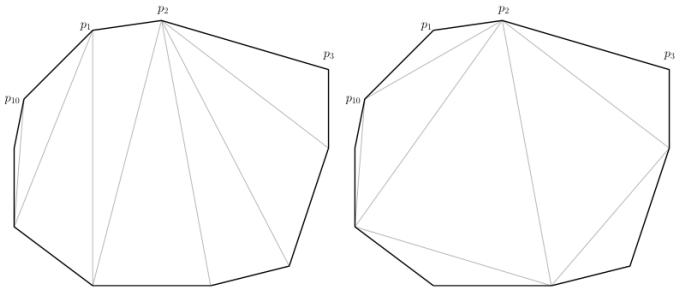
$T[1,1] = 0$

for $k = 2$ to n do:

$$T[1,k] = \min_{j=1 \ldots k-1} \{T[1,j] + C[j,k]\}$$

The running time is $O(n^2)$.

Problem 3. Suppose we are given a convex n-sided polygon P (a polygon is called convex if the angle of each interior corner is less than 180 degrees). A *triangulation* of P is adding diagonals connecting the vertices of P so that no two diagonals intersect each other (except at a corner if they share that corner) and every interior face (region) is a triangle. The following pictures show two different triangulations of a convex polygon (with $n = 10$).



The cost of a triangulation is the sum of the lengths of the diagonals added. For any two corners p_i and p_j of P (with coordinates x_i, y_i and x_j, y_j) assume we have a method that can compute the length of the diagonal between p_i and p_j in constant time and also compare these lengths in constant time. Present an efficient algorithm for computing a minimum cost triangulation of P. Explain why the algorithm is correct and do the analysis of the running time as well.

```
procedure polygon_triangulation(1, n)
for i ← 1 to n-2 do
    m[i, i+1] ← 0
    m[i, i+2] ← 0
m[n-1, n] ← 0
for shift ← 3 to n do:
    for i ← 1 to n-shift do:
        j ← i + shift
        m[i, j] ← ∞
        for k ← i+1 to j-1 do
            if k > i+1 and k < j-1 then
                new ← m[i, k] + m[k+1, j] + dik + dkj
            else if k = i+1 then
                new ← m[i, k] + m[k+1, j] + dkj
            else
```

```
else
    new ← m[i,k] + m[k+1,j] + d_ik
if new < m[i,j] then
    m[i,j] ← new
    s[i,j] ← k
return m[1,n]
```

$O(n^3)$

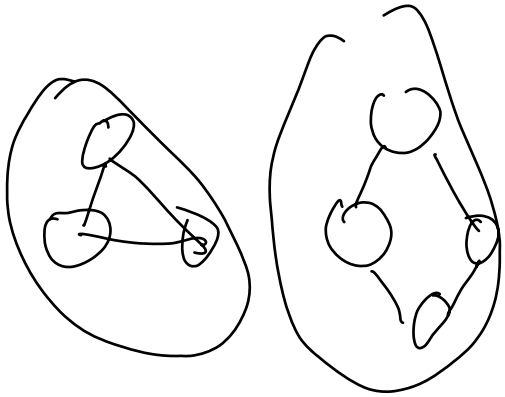
Problem 4. Suppose we are given an n-node rooted tree T, such that each node v in T is given a weight $w(v) \in \mathbb{R}^+$. An independent set of T is a subset S of the nodes of T such that no node in S is a child or parent of any other node in S. Design an efficient dynamic programming algorithm to find the maximum-weight independent set of the nodes in T, where the weight of a set of nodes is simply the sum of the weights of the nodes in that set. What is the running time of your algorithm?

```
procedure max_ind_set(V)
  if V has no children:
      c[V,1] = w(V)
      c[V,2] = 0
  else
      let u1, u2, ..., u_d(V) be children of V
      for i ← 1 to d(V) do
          max_ind_set(u_i)
      c[V,1] = w(V)
      c[V,2] = 0
      for i ← 1 to d(V) do
          c[V,1] = c[V,1] + c[u_i,2]
          c[V,2] = c[V,2] + c[u_i,1]
```

$O(n)$

Problem 5. A company named $RT\&T$ has a network of n switching stations connected by m high-speed communication links. Each customer's phone is directly connected to one station in his or her area. The enginners of $RT\&T$ have developed a prototype video-phone system that allows two customers to see each other during a phone call. In order to have acceptable image quality, however, the number of links used to transmit video signals between the two parties cannot exceed 4. Suppose that $RT\&T$'s network is represented by a graph. Design an efficient algorithm that given this graph, computes for each station, the set of stations it can reach using no more than 4 links.

run BFS start at v, as soon as reach the level 4, stop BFS search.



```
procedure DFS-visit(u, v):
if i > 0 then
    mark v as explored
    for each u ∈ Adj[v] do
        if u is not explored then
            mark u, v as a tree edge
            p[u] ← v
            DFS_visit[u, i-1)
```

Problem 6. The square of an undirected graph $G(V, E)$ is a new graph $G^2(V, E')$ where edge $uv \in E'$ if and only if the distance between u and v in G is at most 2, i.e. either $uv \in E$ or there is a $w \in V$ such that $uw \in E$ and $wv \in E$. Describe efficient algorithms for computing G^2 for each of the following two cases:

i) G is represented in adjacency list,

ii) G is represented in adjacency matrix.

Analyze the running time of your algorithm.

---

i

```
procedure G-square(V[G], E[G])
    V[G²] ← V[G]
    for each u ∈ V[G]
        for each v ∈ adj[u]
            for each w ∈ adj[v]
                E[G²] ← {(u, w)} ∪ E[G²]
```

$O(V^3)$

```
// procedure adj-matrix
for i=1 to V
  for j=1 to V
    a^L[i][j] = 0
      for k=1 to V
        if (g[i][k] == 1 and g[k][j] == 1)
            a^L[i][j] = 1
            break
        end for
  end for
end for
```

$O(V^3)$

8.a.

```
DFS_visit_print(G, v)
time = time + 1
u.d = time
u.color = gray
for each v ∈ G.adj[u]
    if v.color = white
        print (u,v) is a tree edge
        v.π = u
        DFS_visit_printing(u, v)
    else if v.color = gray
        print (u,v) is a Back edge.
    else
        if v.d > u.d
            print (u,v is a forward edge)
        else
            print (u,v is a cross edge)
```

Problem 7. A digraph $G = (V, E)$ is called a *one-way connected graph* if for every two vertices $u, v \in V$, either there is a path from u to v or there is a path from v to u (but never both, so u and v can never be strongly connected).

a. State a necessary and sufficient condition for a digraph to be one-way connected. Prove its correctness (i.e., prove it is a necessary condition and also a sufficient condition).

Note: Here is an incorrect answer. Consider the condition "every pair of vertices u and v are not strongly connected". It is obviously a necessary condition, since a one-way connected graph cannot have both a path from u to v and a path from v to u. While it is NOT a sufficient condition, since a digraph satisfying this condition may not be a one-way connected graph (e.g., in a digraph with three vertices $1, 2, 3$ and two edges $\langle 1, 2\rangle$ and $\langle 3, 2\rangle$, each pair of vertices are not strongly connected, and the digraph is also not one-way connected since there is no path between 1 and 3).

b. Describe an efficient algorithm to determine whether or not an input digraph $G = (V, E)$ is one-way connected. Analyze the running time of your algorithm. (Hint: Make use of the condition you state in part **a**.)