Surname: .... Name: ....

Group: .... Date: ....

Question	1	2	3	4	5	6	7	Total
Points	15	10	10	15	15	20	15	100
Grade								

Answer the questions in the spaces provided. If you run out of room for an answer, continue on the back of the page.

1. (*15 points*) Directly determine if $f(n) = O(g(n))$, or if $f(n) = \Omega(g(n))$, or if $f(n) = \Theta(g(n))$.

   (a) $f(n) = 2^{2n}, g(n) = 2^n$.

   (b) $f(n) = n^{\log c}, g(n) = c^{\log n}$.

   (c) $f(n) = 8\log(n^n), g(n) = 100\log(n!)$.

   (d) $f(n) = n, g(n) = \log^2 n$.

   (e) $f(n) = n\log n + n, g(n) = \log n + n$.

Solution:

a)We evaluate the limit:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} \frac{(2^n)^2}{2^n} = \lim_{n\to\infty} 2^n = \infty$$

Since the limit is ∞, $f(n)$ grows asymptotically faster than $g(n)$. Therefore, $f(n) = \Omega(g(n))$.

b)We use the identity $a^b = b^{\log_k a}$ for any base k. More simply, we can use the property $x^y = (e^{\ln} x)^y = e^{y\ln x} y = e^{y\ln x}$.

$$f(n) = n^{\log c} = e^{\ln(n^{\log c})} = e^{\log c \cdot \ln n}$$

$$g(n) = c^{\log n} = e^{\ln(c^{\log n})} = e^{\log n \cdot \ln c}$$

Since $\ln n \cdot \ln c = \ln c \cdot \ln n$, we have $f(n) = g(n)$. Therefore, $f(n) = \Theta(g(n))$.

c) First, simplify $f(n)$ using the logarithm property $\log(a^b) = b\log a$:

$$f(n) = 8\log(n^n) = 8n\log n$$

For $g(n)$, we use Stirling's approximation for $\ln(n!)$ which states $\ln(n!) \approx n\ln n - n$. Assuming log is ln (natural logarithm) or any other base, the dominant term is $n\log n$.

$$g(n) = 100\log(n!) \approx 100(n\log n - n) = 100n\log n - 100n$$

Now, we evaluate the limit:

Model A

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{8n\log n}{100n\log n - 100n} = \lim_{n\to\infty} \frac{8n\log n}{100n(\log n - 1)}$$

$$= \lim_{n\to\infty} \frac{8\log n}{100(\log n - 1)} = \lim_{n\to\infty} \frac{8}{100\left(1 - \frac{1}{\log n}\right)} = \frac{8}{100(1-0)} = \frac{8}{100} = \frac{2}{25}$$

Since the limit is a positive finite constant $\frac{2}{25}$, $f(n)$ and $g(n)$ have the same asymptotic growth rate.

Therefore, $f(n) = \Theta(g(n))$.

d)We evaluate the limit:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n}{\log^2 n}$$

This is an indeterminate form $\frac{\infty}{\infty}$, so we can apply L'Hôpital's rule. We assume $\log n$ is $\ln n$.

Applying L'Hôpital's rule once:

$$\lim_{n\to\infty} \frac{\frac{d}{dn}(n)}{\frac{d}{dn}(\ln^2 n)} = \lim_{n\to\infty} \frac{1}{2\ln n \cdot \frac{1}{n}} = \lim_{n\to\infty} \frac{n}{2\ln n}$$

This is still an indeterminate form $\frac{\infty}{\infty}$, so we apply L'Hôpital's rule again:

$$\lim_{n\to\infty} \frac{\frac{d}{dn}(n)}{\frac{d}{dn}(2\ln n)} = \lim_{n\to\infty} \frac{1}{2\cdot\frac{1}{n}} = \lim_{n\to\infty} \frac{n}{2} = \infty$$

Since the limit is ∞, $f(n)$ grows asymptotically faster than $g(n)$.

Therefore, $f(n) = \Omega(g(n))$.

e)We evaluate the limit:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n\log n + n}{\log n + n}$$

Divide both numerator and denominator by n:

$$= \lim_{n\to\infty} \frac{n\log n}{n} + \frac{\frac{\frac{n}{n}}{\log n}}{n} + \frac{n}{n} = \lim_{n\to\infty} \frac{\frac{\log n + 1}{\log n}}{n} + 1$$

We know that $\lim_{n\to\infty} \frac{\log n}{n} = 0$.

$$= \frac{\infty + 1}{0+1} = \frac{\infty}{1} = \infty$$

Since the limit is ∞, $f(n)$ grows asymptotically faster than $g(n)$.

Therefore, $f(n) = \Omega(g(n))$.

2. (*10 points*) Solve the recurrence relation: for $n \geq 2$, $f(n) = 5f(n-1) - 6f(n-2)$; $f(0) = 1$; $f(1) = 0$.

Model A

Solution:

For the recurrence relation $f(n) = 5f(n-1) - 6f(n-2)$, the characteristic equation is:

$$x^2 - 5x + 6 = 0$$

Factoring: $(x-2)(x-3) = 0$, so $x_1 = 2$ and $x_2 = 3$.

The general solution is: $f(n) = c_1 \cdot 2^n + c_2 \cdot 3^n$

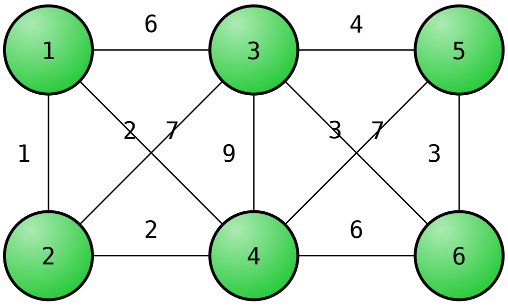
Using initial conditions $f(0) = 1$ and $f(1) = 0$:

$$\begin{cases} c_1 + c_2 = 1 \\ 2c_1 + 3c_2 = 0 \end{cases}$$

Solving: $c_2 = -2$ and $c_1 = 3$.

Therefore: $f(n) = 3 \cdot 2^n - 2 \cdot 3^n$

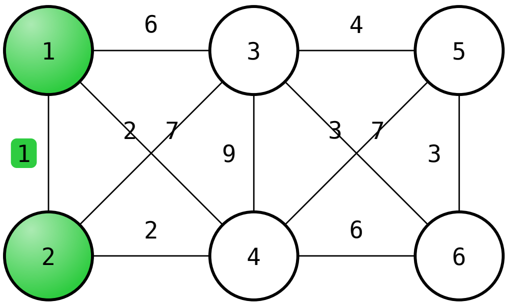
3. (*10 points*) Using Prim's algorithm, find the minimum spanning tree of the graph below.



Solution:

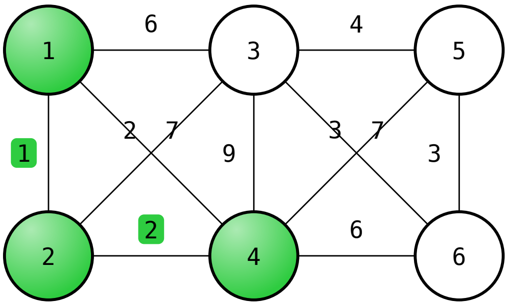
Prim's Algorithm Steps

Step 1: Start with node 1. Available edges: 1-2(1), 1-3(6), 1-4(7). Choose 1-2 with weight 1.

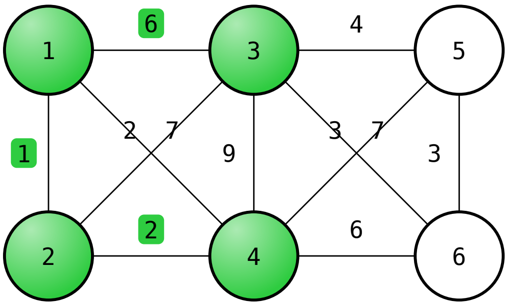


Step 2: MST = {1,2}. Available edges: 1-3(6), 1-4(2), 2-4(2), 2-3(7). Choose 2-4 with weight 2.

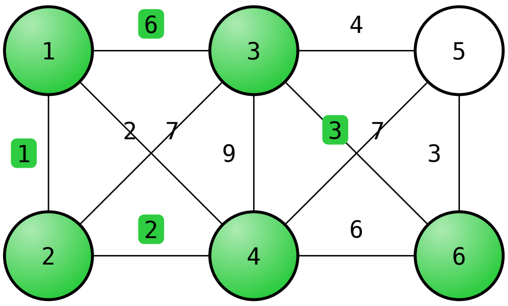
Model A



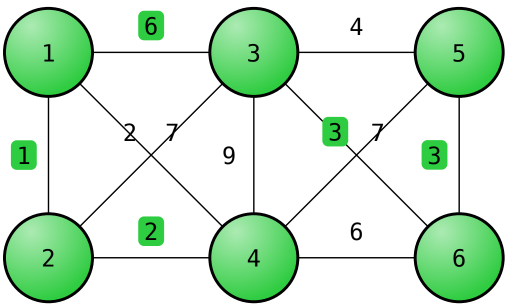
Step 3: MST = {1,2,4}. Available edges: 1-3(6), 2-3(7), 4-6(6), 4-5(3). Choose 1-3 with weight 6.



Step 4: MST = {1,2,3,4}. Available edges: 3-5(4), 3-6(3), 4-6(6), 4-5(7). Choose 3-6 with weight 3.



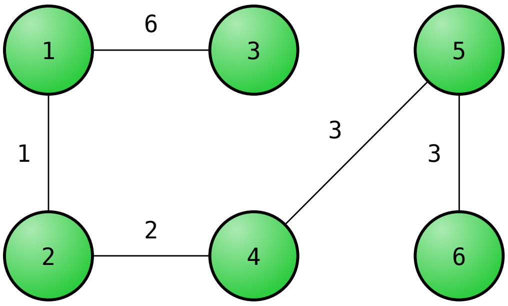
Step 5: MST = {1,2,3,4,6}. Available edges: 3-5(4), 4-5(7), 5-6(3). Choose 5-6 with weight 3.



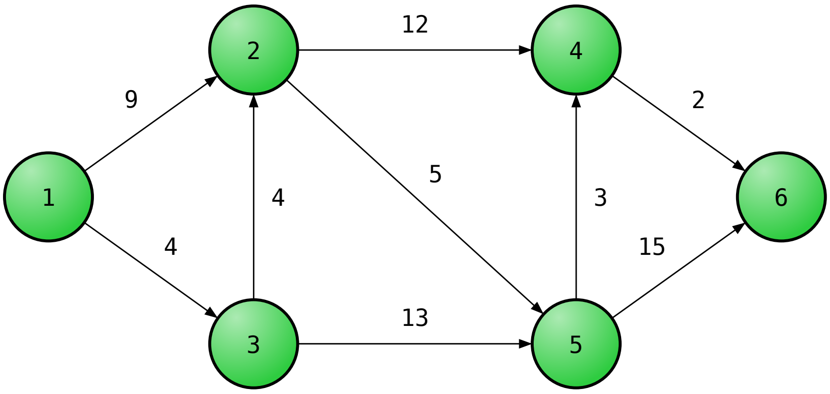
Step 6: All nodes included. MST complete.

Final MST:

Model A



4. (*15 points*) Using Dijkstra's algorithm, solve the single-source shortest path problem for the graph below, with the source node set to 1.



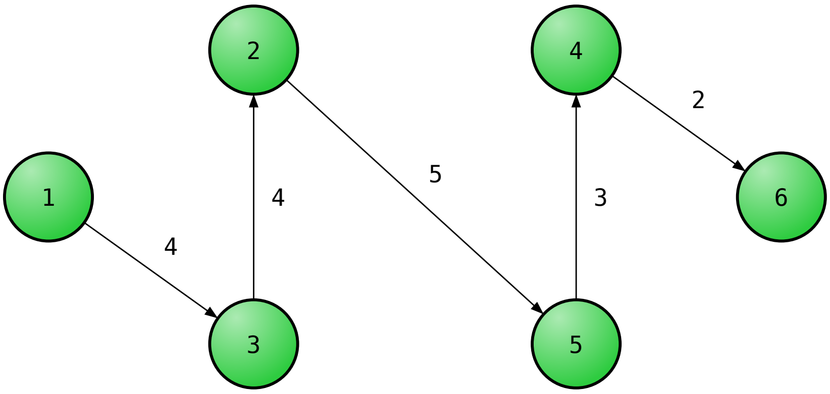
Solution:

Dijkstra's Algorithm Steps

Step	**1**	**2**	**3**	**4**	**5**	**6**
1	0	∞	∞	∞	∞	∞
2		9	4	∞	∞	∞
3		8		∞	17	∞
4				20	13	∞
5				16		28
6						18

Final Shortest Distances

Model A



5. (*15 points*) Use dynamic programming to solve the $0-1$ knapsack problem. Given that the knapsack capacity is 22, and the volumes of 5 items are $3, 5, 7, 8, 9$ respectively, with corresponding values of $4, 6, 7, 9, 10$. Find the maximum value of the knapsack and the selected items.

Solution:

It.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	0	0	0	4	4	6	6	6	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
3	0	0	0	4	4	6	6	7	10	10	11	11	13	13	13	17	17	17	17	17	17	17	17
4	0	0	0	4	4	6	6	7	10	10	11	13	13	15	15	17	19	19	20	20	22	22	22
5	0	0	0	4	4	6	6	7	10	10	11	13	14	15	16	17	19	20	20	21	23	23	25

Therefore, the maximum value of items we can select is 25, and the items chosen are 2, 4, and 5.

6. Find the matrix chain multiplication for the following 5 matrices: $M1(4 \times 5)$; $M2(5 \times 4)$; $M3(4 \times 6)$; $M4(6 \times 4)$; $M5(4 \times 5)$.

(a) (*10 points*) Using either a textual description or pseudocode, outline the dynamic programming algorithm for the problem.

Solution:

```
Matrix-Chain(p, n):
  for i ← 1 to n do m[i][i] ← 0;
  for l ← 2 to n do // l is length of sub-chain
      for i ← 1 to n - l + 1 do
          j ← i + l - 1;
          m[i][j] ← ∞;
          for k ← i to j - 1 do
              q ← m[i][k] + m[k+1][j] + p[i-1] * p[k] * p[j];
              if q < m[i][j] then
                  m[i][j] ← q;
                  s[i][j] ← k;
  return m and s;
```

(b) (*10 points*) Describe how this algorithm is used to solve the problem, and present the final results.

Solution:

Table of multiplication costs:

	1	2	3	4	5
1	0	80	176	240	320
2		0	120	176	276
3			0	96	176
4				0	120
5					0

Table of optimal splits:

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

Therefore, the optimal parenthesization for this matrix chain is $((M_1 M_2)(M_3 M_4))M_5$, and the minimum number of multiplications required is 320.

7. (*15 points*) Let A be a sequence of n numbers. An element x in A is called an "approximate median" if the number of elements less than x is at least $\frac{n}{3}$, and the number of elements greater than x is also at least $\frac{n}{3}$. Design an algorithm to find an approximate median of A. Explain the design idea of your algorithm and its worst-case time complexity.

Solution:

An approximate median means the element x's rank falls between $\frac{n}{3}$ and $\frac{2n}{3}$. In simpler terms, x is an element located in the middle third of the sorted array.

Therefore, any element in the middle third of a sorted array will satisfy the approximate median condition. We can use QuickSort to completely sort the array, then select any element from the valid range.

Algorithm: QuickSort Approach

```
void QSort(Elem A[], int p, int q) {
  if (p >= q) return;
  Elem pivot = A[p];
  int m = partition(A, p, q, pivot);
  QSort(A, p, m - 1);
  QSort(A, m + 1, q);
}
```

Model A

```
int partition(Elem A[], int p, int q, Elem x) {
  int i = p;
  for (int j = p + 1; j <= q; ++j) {
      if (A[j] <= x) {
          ++i;
          swap(A[i], A[j]);
      }
  }
  swap(A[p], A[i]);
  return i;
}
```

Then, just return any element from the range $A\left[\frac{n}{3}\right]$ to $A\left[\frac{2n}{3}\right]$.

The worst-case time complexity of this algorithm is $O(n \log n)$.

Model A