Problem #1 **Name:** Alexander Michelbrink

A.)_____

Following the Greedy algorithm for the fractional knapsack problem as discussed in class, we first must calculate the value index v_i of each item.

$$v_a = \frac{12}{3} = 4$$
$$v_b = \frac{12}{6} = 2$$
$$v_c = \frac{9}{6} = 1.5$$
$$v_d = \frac{5}{1} = 5$$
$$v_e = \frac{5}{2} = 2.5$$
$$v_f = \frac{10}{10} = 1$$
$$v_g = \frac{9}{3} = 3$$

Next, we consider the items in the order of highest value index to lowest value index. We take as much of the highest value item as possible (without exceeding W = 11 lbs), add its weight to the current weight w (starting at w = 0 lbs), and then move onto the next item. We are done once we take only a fraction of an item that causes w = W.

Highest-to-Lowest Value Index: $v_d, v_a, v_g, v_e, v_b, v_c, v_f$

Steps…

1. 1 lbs of Item D (w = 1)
2. 3 lbs of Item A (w = 4)
3. 3 lbs of Item G (w = 7)
4. 2 lbs of Item E (w = 9)
5. 2 lbs of Item B (w = 11)

Solution…

The optimal solution involves taking all of Item D, Item A, Item G, and Item E, while only taking 2 lbs of Item B (the prevent w from exceeding W = 11 lbs).

B.)_____

We can model the dynamic programming algorithm for the 0-1 knapsack problem as discussed in class using the following table. I got the idea for the visualization of this table from the video cited below, although I used the equation for B[k,w] from class to fill in its values.

Items:
A: w = 3, benefit = 12
B: w = 6, benefit = 12
C: w = 9, benefit = 9
D: w = 1, benefit = 5
E: w = 2, benefit = 5
F: w = 10, benefit = 10
G: w = 3, benefit = 9

	weights											
items	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	12	12	12	12	12	12	12	12	12
2	0	0	0	12	12	12	12	12	12	24	24	24
3	0	0	0	12	12	12	12	12	12	24	24	24
4	0	5	5	12	17	17	17	17	17	24	29	29
5	0	5	5	12	17	17	22	22	22	24	29	29
6	0	5	5	12	17	17	22	22	22	24	29	29
7	0	5	5	12	17	17	22	26	26	31	31	31

Solution…

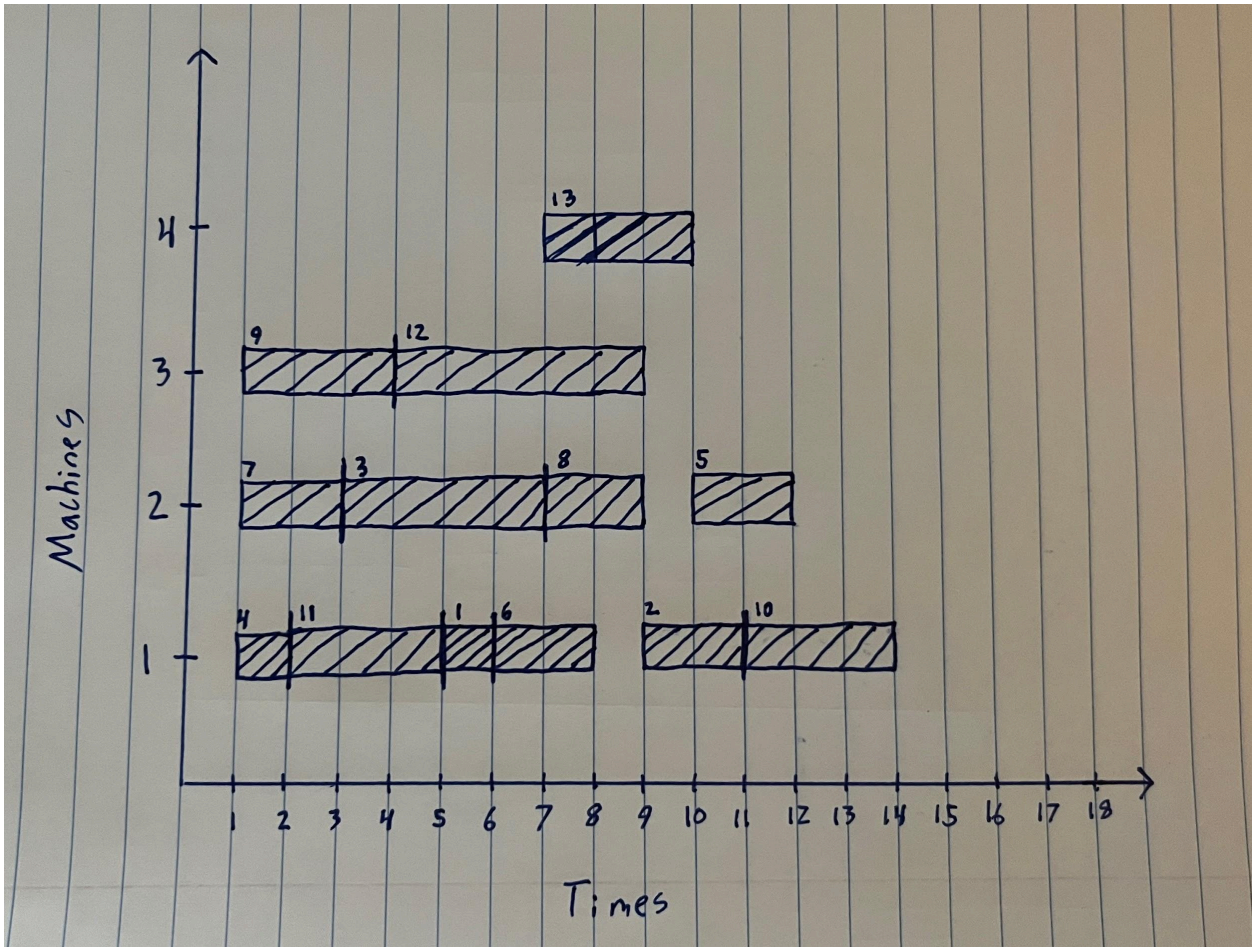
The optimal solution involves taking items A, D, E, and G with a resultant benefit of 31. This uses 9lbs of the 11lbs knapsack.

Sources

Abdul Bari. *4.5 0/1 Knapsack - Two Methods - Dynamic Programming*. https://www.youtube.com/watch?v=nLmhmB6NzcM

Problem #2

Following the TaskSchedule() algorithm gives the following schedule that uses the minimum number of machines possible. In this case, 4 machines is the minimal amount.



Problem #3

A.)_____

Notations: f_i: The finish time of task ***i***.
s_i: The start time of task ***i***.
U: A collection storing the tasks that will not fit onto the single machine in the optimized solution.

The Algorithm

```
Algorithm SingleMachineScheduling(T)
    while (T ≠ ϕ) do
        remove from T the task i with the smallest finish time f_i
        if (there's not a task conflicting with i on the machine)
            schedule task i on the machine
        else
            store task i in U
```

Explanation

The algorithm loops through each task ***i*** (in order of smallest finish time to largest finish time) and assigns the task to the machine if there are currently no tasks on the machine that would conflict with it. If not, it puts the task into a separate collection ***U*** for use later (possibly involving scheduling on a different machine).

In the case of two tasks with the same finish time, either task could be scheduled without affecting the total amount of tasks that could fit onto the single machine. Therefore, the choice between the two tasks can be determined based purely on their relative order in the collection ***T***.

The runtime of this algorithm would be O(*nlog(n)*), where there are ***n*** tasks in ***T***. This runtime is ensured by implementing ***T*** and ***U*** as minimum priority queues. This leads to the removal of the shortest task from ***T*** taking O(*log(n)*) time and the inserting of the discarded task into ***U*** also taking O(*log(n)*) time. Assuming scheduling the task on the machine takes O(*1*) time, the while-loop has a runtime of O(*nlog(n)*).

B.)_____

The correctness of the above algorithm can be proved by contradiction.

Suppose the algorithm above is not correct, and there exists a different scheduling of tasks on the single machine that supports an additional task. In other words, SingleMachineScheduling(***T***) produces a schedule with ***k*** tasks, but there exists a schedule with ***k+1*** tasks. This means that, at some point, choosing a task other than the one with the smallest end-time was the better choice.

The only way this "better choice task" would have led to ***k+1*** (or more) tasks being scheduled is if it conflicted with fewer tasks then the task that currently has the smallest finish time that would be allocated to the machine instead. This is not possible, due to the following:

> The only way any tasks could conflict with a task with a shorter-or-equal endtime (task 1) and not conflict with a task with a longer or equal endtime (task 2) is if they fell within the timeframe (s_1 , s_2). Any other tasks conflicting with 1 would have to fall into the timeframe (s_2 , f_1), meaning they would also have to conflict with task 2 (since $f_1 \leq f_2$).
>
> It is not possible for there to be any conflicting task ***n*** in the timeframe (s_1 , s_2) since this task would have been scheduled prior to task 1, causing task 1 to be skipped. The task would have had to have been scheduled prior to task 1 since it must have a shorter-or-equal finish time than task 1 ($s_2 < f_1 \; and \; f_n \leq s_2 \Rightarrow \; f_n < f_1$).

Therefore, when choosing between conflicting tasks, the task with the shortest endtime must have an equal or smaller amount of conflicts than any of the other possible tasks with longer or equal finish times. Therefore, choosing the greedy choice of the task with the smallest finish time is always the optimal option and leads to the optimal solution. This makes the above algorithm correct.

Problem #4

a.)

$T(n) = 2T(\frac{n}{2}) + log(n)$

- a = 2, b = 2
- Special function $n^{log_b a} = n^1 = n$, so $f(n) = log(n)$ is polynomially less than the special function which leads to case #1…
- $f(n)\ is\ O(n^{log_b a - \varepsilon}) \Rightarrow log(n)\ is\ O(n^{1-\varepsilon}) \Rightarrow log(n)\ is\ O(\sqrt{n})$ for $\varepsilon = \frac{1}{2}$.
    - This verifies case #1, meaning…
    - $T(n)\ is\ \Theta(n^{log_b a})$

Solution: $T(n)\ is\ \Theta(n)$

b.)

$T(n) = 8T(\frac{n}{2}) + n^2$

- a = 8, b = 2
- Special function $n^{log_b a} = n^3$, so $f(n) = n^2$ is polynomially less than the special function which leads to case #1…
- $f(n)\ is\ O(n^{log_b a - \varepsilon}) \Rightarrow n^2\ is\ O(n^{3-\varepsilon}) \Rightarrow n^2\ is\ O(n^2)$ for $\varepsilon = 1$.
    - This verifies case #1, meaning…
    - $T(n)\ is\ \Theta(n^{log_b a})$

Solution: $T(n)\ is\ \Theta(n^3)$

c.)

$T(n) = 7T(\frac{n}{3}) + n$

- a = 7, b = 3
- Special function $n^{log_b a} = n^{log_3 7} \approx n^{1.77}$, so $f(n) = n$ is polynomially less than the special function which leads to case #1…
- $f(n)\ is\ O(n^{log_b a - \varepsilon}) \Rightarrow n\ is\ O(n^{log_3 7 - \varepsilon}) \Rightarrow n\ is\ O(n)$ for $\varepsilon = (log_3 7) - 1$.
    - This verifies case #1, meaning…
    - $T(n)\ is\ \Theta(n^{log_b a})$

Solution: $T(n)\ is\ \Theta(n^{log_3 7})$, or $T(n)\ is\ \Theta(n^{1.77})$ when approximating.

d.)_____

$T(n) = 4T(\frac{n}{2}) + n^2$

- a = 4, b = 2
- Special function $n^{log_b a} = n^2$, so $f(n) = n^2$ is asymptotically close to the special function which leads to case #2…
- $f(n)\ is\ \Theta(n^{log_b a} log^k(n)) \Rightarrow n^2\ is\ \Theta(n^2 log^k(n)) \Rightarrow n^2\ is\ \Theta(n^2)$ for $k = 0$.
  - This verifies case #2, meaning…
  - $T(n)\ is\ \Theta(n^{log_b a} log^{k+1} n)$

Solution: $T(n)\ is\ \Theta(n^2 log(n))$

e.)_____

$T(n) = 3T(\frac{n}{2}) + n^2$

- a = 3, b = 2
- Special function $n^{log_b a} = n^{log_2 3} \approx 1.58$, so $f(n) = n^2$ is polynomially greater than the special function which leads to case #3…
- $f(n)\ is\ \Omega(n^{log_b a + \varepsilon}) \Rightarrow n^2\ is\ \Omega(n^{log_2 3 + \varepsilon}) \Rightarrow n^2\ is\ \Omega(n^{log_2 3 + \frac{1}{4}})$ for $\varepsilon = \frac{1}{4}$, AND $af(\frac{n}{b}) \leq \delta f(n) \Rightarrow 3(\frac{n}{2})^2 \leq \delta n^2 \Rightarrow \frac{3}{4}n^2 \leq \frac{3}{4}n^2$ for $\delta = \frac{3}{4}$
  - This verifies case #3, meaning…
  - $T(n)\ is\ \Theta(f(n))$

Solution: $T(n)\ is\ \Theta(n^2)$

Problem #5

$A_1 = 10 \times 5,\ A_2 = 5 \times 2,\ A_3 = 2 \times 20,\ A_4 = 20 \times 12,\ A_5 = 12 \times 4,\ A_6 = 4 \times 60$
$d_1 = 10,\ d_2 = 5,\ d_3 = 2,\ d_4 = 20,\ d_5 = 12,\ d_6 = 4,\ d_7 = 60$

$N_{i,i}$ ———

- $N_{1,1} = N_{2,2} = N_{3,3} = N_{4,4} = N_{5,5} = N_{6,6} = 0$

$N_{i,i+1}$ ———

- $N_{1,2} = N_{1,1} + N_{2,2} + d_1 d_2 d_3 = 0 + 0 + 100 = 100,\ k = 1$
- $N_{2,3} = N_{2,2} + N_{3,3} + d_2 d_3 d_4 = 0 + 0 + 200 = 200,\ k = 2$
- $N_{3,4} = N_{3,3} + N_{4,4} + d_3 d_4 d_5 = 0 + 0 + 480 = 480,\ k = 3$
- $N_{4,5} = N_{4,4} + N_{5,5} + d_4 d_5 d_6 = 0 + 0 + 960 = 960,\ k = 4$
- $N_{5,6} = N_{5,5} + N_{6,6} + d_5 d_6 d_7 = 0 + 0 + 2880 = 2880,\ k = 5$

$N_{i,i+2}$ ———

- $N_{1,3} = min\{N_{1,1} + N_{2,3} + d_1 d_2 d_4 = 0 + 200 + 1000 = 1200,$
  $N_{1,2} + N_{3,3} + d_1 d_3 d_4 = 100 + 0 + 400 = 500\}$
  $= 500,\ k = 2$
- $N_{2,4} = min\{N_{2,2} + N_{3,4} + d_2 d_3 d_5 = 0 + 480 + 120 = 600,$
  $N_{2,3} + N_{4,4} + d_2 d_4 d_5 = 200 + 0 + 1200 = 1400\}$
  $= 600,\ k = 2$
- $N_{3,5} = min\{N_{3,3} + N_{4,5} + d_3 d_4 d_6 = 0 + 960 + 160 = 1120,$
  $N_{3,4} + N_{5,5} + d_3 d_5 d_6 = 480 + 0 + 96 = 576\}$
  $= 576,\ k = 4$
- $N_{4,6} = min\{N_{4,4} + N_{5,6} + d_4 d_5 d_7 = 0 + 2880 + 14400 = 17280,$
  $N_{4,5} + N_{6,6} + d_4 d_6 d_7 = 960 + 0 + 4800 = 5760\}$
  $= 5760,\ k = 5$

$N_{i,i+3}$ ———

- $N_{1,4} = min\{N_{1,1} + N_{2,4} + d_1 d_2 d_5 = 0 + 600 + 600 = 1200,$
  $N_{1,2} + N_{3,4} + d_1 d_3 d_5 = 100 + 480 + 240 = 820,$
  $N_{1,3} + N_{4,4} + d_1 d_4 d_5 = 500 + 0 + 2400 = 2900\}$
  $= 820,\ k = 2$

- $N_{2,5} = min\{N_{2,2} + N_{3,5} + d_2 d_3 d_6 = 0 + 576 + 40 = 616,$
  $N_{2,3} + N_{4,5} + d_2 d_4 d_6 = 200 + 960 + 400 = 1560,$
  $N_{2,4} + N_{5,5} + d_2 d_5 d_6 = 600 + 0 + 240 = 840\}$
  $= 616,\ k = 2$
- $N_{3,6} = min\{N_{3,3} + N_{4,6} + d_3 d_4 d_7 = 0 + 5760 + 2400 = 8160,$
  $N_{3,4} + N_{5,6} + d_3 d_5 d_7 = 480 + 2880 + 1440 = 4800,$
  $N_{3,5} + N_{6,6} + d_3 d_6 d_7 = 576 + 0 + 480 = 1056\}$
  $= 1056,\ k = 5$

$N_{i,i+4}$ ———

- $N_{1,5} = min\{N_{1,1} + N_{2,5} + d_1 d_2 d_6 = 0 + 616 + 200 = 816,$
  $N_{1,2} + N_{3,5} + d_1 d_3 d_6 = 100 + 576 + 80 = 756,$
  $N_{1,3} + N_{4,5} + d_1 d_4 d_6 = 500 + 960 + 800 = 2260,$
  $N_{1,4} + N_{5,5} + d_1 d_5 d_6 = 820 + 0 + 480 = 1300\}$
  $= 756,\ k = 2$
- $N_{2,6} = min\{N_{2,2} + N_{3,6} + d_2 d_3 d_7 = 0 + 1056 + 600 = 1656,$
  $N_{2,3} + N_{4,6} + d_2 d_4 d_7 = 200 + 5760 + 6000 = 11960,$
  $N_{2,4} + N_{5,6} + d_2 d_5 d_7 = 600 + 2880 + 3600 = 7080,$
  $N_{2,5} + N_{6,6} + d_2 d_6 d_7 = 616 + 0 + 1200 = 1816\}$
  $= 1656,\ k = 2$

$N_{i,i+5}$ ———

- $N_{1,6} = min\{N_{1,1} + N_{2,6} + d_1 d_2 d_7 = 0 + 1656 + 3000 = 4656,$
  $N_{1,2} + N_{3,6} + d_1 d_3 d_7 = 100 + 1056 + 1200 = 2356,$
  $N_{1,3} + N_{4,6} + d_1 d_4 d_7 = 500 + 5760 + 12000 = 18260,$
  $N_{1,4} + N_{5,6} + d_1 d_5 d_7 = 820 + 2880 + 7200 = 10900,$
  $N_{1,5} + N_{6,6} + d_1 d_6 d_7 = 756 + 0 + 2400 = 3156\}$
  $= 2356,\ k = 2$

Solution _____

Optimal Sequence: $(A_{1,2})(A_{3,6})$

$((A_1)(A_2))((A_{3,5})(A_6))$

$((A_1)(A_2))(((A_{3,4})(A_5))(A_6))$

$((A_1)(A_2))((((A_3)(A_4))(A_5))(A_6))$

Optimal Multiplication Numbers Table

	1	**2**	**3**	**4**	**5**	**6**
6	2356	1656	1056	5760	2880	0
5	756	616	576	960	0	
4	820	600	480	0		
3	500	200	0			
2	100	0				
1	0					

k-Index Table

	1	**2**	**3**	**4**	**5**	**6**
6	2	2	5	5	5	
5	2	2	4	4		
4	2	2	3			
3	2	2				
2	1					
1						