THE UNIVERSITY of ADELAIDE

Optimisation & Operations Research

Haide College, Spring Semester

Assignment 3 (5%)

Due: 30 April, 23:59

This assignment covers Topic 4 (Integer Programming Problems).

10 marks

1. Recall from the course notes that the knapsack problem involves choosing which items to put in a knapsack (or backpack). Each item has value, but you have a limited volume in the pack.

   Consider the following linear program representing a knapsack problem for six items with total value z and maximum volume 35.

$$\begin{array}{lrcrcrcrcrcrcl} \text{max.} & z & = & 19x_1 & + & 22x_2 & + & 30x_3 & + & 37x_4 & + & 11x_5 & + & 42x_6, \\ \text{s.t.} & & & 7x_1 & + & 6x_2 & + & 11x_3 & + & 13x_4 & + & 4x_5 & + & 13x_6 & \leq & 35, \end{array}$$

   with $x_i = 0, 1$, for $i = 1, \ldots, 6$.

2 a) Relax the integral constraint to $x_i \geq 0$ (and real). Solve the relaxed version of the linear program either by inspection or with MATLAB.

Solution: Using the notation value v_i and weight w_i (as in $\max v = \sum_i v_i x_i$, s.t. $\sum_i w_i x_i \leq w$), we have

i	1	2	3	4	5	6
v_i	19	22	30	37	11	42
w_i	7	6	11	13	4	13
v_i/w_i	2.71	3.67	2.73	2.85	2.75	3.23

Since variable x_2 has the best value to weight (v_i/w_i) ratio and there is no integer or binary requirement, we should make x_2 as large as possible. This gives

- $x_2^* = 35/6 = 5.8333$, and all other $x_i^* = 0$.
- Then $z^* = 22 \times x_2^* = 128.3333$

```
%% OOR 2025 - Assignment 3, Question 1(a)
% Mike Chen, April 2025
% Input the knapsack problem ...
f = -[19 22 30 37 11 42];
A = [7 6 11 13 4 13];
b = 35;
Aeq = [];
beq = [];
% ... with only non-negativity constraints
lb = zeros(1,6);
ub = [];
% Solve with linprog
[x_1a,fval_1a] = linprog(f,A,b,Aeq,beq,lb,ub);
```

Output:

```
x_1a =
```

```
         0
    5.8333
         0
         0
         0
         0

fval_2a =

 -128.3333
```

1 b) Relax the integral constraint to $0 \leq x_i \leq 1$ (and real). Solve the relaxed problem with MATLAB and submit via MATLAB Grader. You do **not** need to include your code.

Solution:

```
%% OOR 2054 - Assignment 3, Question 1(b)
% Mike Chen, April 2025
% Input the knapsack problem ...
f = -[19 22 30 37 11 42];
A = [7 6 11 13 4 13];
b = 35;
Aeq = [];
beq = [];
% ... with constraints that 0 <= x_i <= 1 for all x_i.
lb = zeros(1,6);
ub = ones(1,6);
% Solve with linprog
[x_1b,fval_1b] = linprog(f,A,b,Aeq,beq,lb,ub);
```

Output:

```
x_1b =

         0
    1.0000
         0
    1.0000
    0.7500
    1.0000

fval_1b =

 -109.2500
```

1 c) Now solve the true binary integer linear program with MATLAB and submit via MATLAB Grader. You do **not** need to include your code.

Solution:

```
%% OOR 2025 - Assignment 3, Question 1(c)
% Mike Chen, April 2025
% Input the knapsack problem ...
```

```
f = -[19 22 30 37 11 42];
A = [7 6 11 13 4 13];
b = 35;
Aeq = [];
beq = [];
% ... with constraints that all x_i = 0 or 1.
lb = zeros(1,6);
ub = ones(1,6);
intcon = 1:6;
% Solve with intlinprog
[x_1c,fval_1c] = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub);
```

Output:

```
x_2c =

     0
     1
     1
     0
     1
     1

fval_2c =

  -105
```

2 d) State the solutions (variables and objectives) you found in 1(b) and 1(c).
Comment on the relative size of the objective function for each solution from 1(a)–(c), and discuss why you would expect the value of the objective to change as it does.

Solution: In the above questions the solutions we found were:

- **Q1(a):** $\mathbf{x}^* = (0, 5.8333, 0, 0, 0, 0)$, $z^* = 128.3333$;
- **Q1(b):** $\mathbf{x}^* = (0, 1, 0, 1, 0.75, 1)$, $z^* = 109.25$;
- **Q1(c):** $\mathbf{x}^* = (0, 1, 1, 0, 1, 1)$, $z^* = 105$.

As we proceed from (a) to (b) to (c), the feasible region becomes more restricted, and so the value of z decreases, as we would expect.

4 e) Now apply the greedy algorithm from the course notes to the problem. You should do this by hand, rather than in MATLAB, and include all working in your submission.
How does the greedy solution compare to the optimal solution?

Solution: For reference, here is the value-to-weight table again from Q1(a):

i	1	2	3	4	5	6
v_i	19	22	30	37	11	42
w_i	7	6	11	13	4	13
v_i/w_i	2.71	3.67	2.73	2.85	2.75	3.23

Include items in v_i/w_i order (provided they do not exceed weight):

Item	**Ratio**	**Included**	**Weight cumulative**
2	3.66	Y	6
6	3.23	Y	6 + 13=19
4	2.85	Y	19 + 13 = 32
5	2.75	N	32
3	2.73	N	32
1	2.71	N	32

Or equivalently,

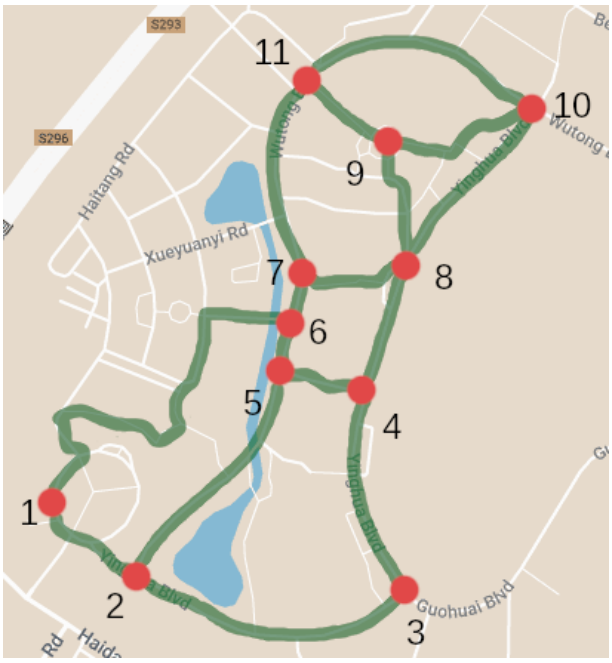
1. Add x_2 ($v_2/w_2 = 3.66$). Current weight $= 6$.
2. Add x_6 ($v_6/w_6 = 3.23$). Current weight $= 19$.
3. Add x_4 ($v_4/w_4 = 2.85$). Current weight $= 32$.
4. Cannot add x_5, x_3 or x_1 without violating weight ≤ 35 constraint.

That gives a solution of $\mathbf{x}^* = (0, 1, 0, 1, 0, 1)$ with $z^* = 101$.
This is a smaller value than the optimal solution of $z^* = 105$ from Q1(c). Notice that the total weight here is 32, versus total weight 34 for the optimal soltuion.

---

10 marks

2. In this question you will explore how mapping apps (Baidu or Gaode, for example) find the shortest routes between locations. Consider the below map of the OUC campus.



Eleven locations have been labelled with numbers, for instance the Haide College (Xingzhi) Building is labelled "1".

A visitor would like to find the shortest routes from the Haide College building to the other locations marked on the map (location 2 to 11). Assume they will only walk between these locations via the footpaths and roads marked in green.

The distances between the locations (along the green footpaths/roads) are given in the table below. **Note it is possible to travel between locations in either direction (for example** $4 \to 5$ **or** $5 \to 4$**), even though only one direction listed below.**

From	To	Distance (m)
1	2	160
1	6	590
2	3	400
2	5	350
3	4	280
4	5	120
4	8	190
5	6	70

From	To	Distance (m)
6	7	40
7	8	150
7	11	290
8	9	170
8	10	260
9	10	220
9	11	110
10	11	350

Apply a greedy heuristic from the course notes to find the shortest routes from the Haide College building (location 1) to the other 10 marked locations.

Include all working in your submission.

As part of your submission include a table showing the shortest route to each destination and the length of the route. (**Hint:** A similar table is given in the solution for Tutorial 4, Question 3).

Solution:
Apply Djikstra's algorithm to find shortest path from 1 to each of the other destinations. Here is a summary of the iterations from Djikstra's algorithm (see below for

full working for each iteration):

Iteration	Added to S	D
Initialise	1	$(0, \mathbf{160}, -, -, -, \mathbf{590}, -, -, -, -, -)$
1	2	$(0, 160, \mathbf{560}, -, \mathbf{510}, 590, -, -, -, -, -)$
2	5	$(0, 160, 560, \mathbf{630}, 510, \mathbf{580}, -, -, -, -, -)$
3	3	$(0, 160, 560, 630, 510, 580, -, -, -, -, -)$
4	6	$(0, 160, 560, 630, 510, 580, \mathbf{620}, -, -, -, -)$
5	7	$(0, 160, 560, 630, 510, 580, 620, \mathbf{770}, -, -, \mathbf{910})$
6	4	$(0, 160, 560, 630, 510, 580, 620, 770, -, -, 910)$
7	8	$(0, 160, 560, 630, 510, 580, 620, 770, \mathbf{940}, \mathbf{1030}, 910)$
8	11	$(0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$
9	9	$(0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$
10 (stop)	10	$(0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$

Looking back at our working we can spot that the predecessor nodes are

Node	**Predecessor**	**Iteration**
1	-	-
2	1	Initialisation
3	2	1
4	5	2
5	2	1
6	5	2
7	6	4
8	7	5
9	8	7
10	8	7
11	7	5

Construct routes and summarise solution to give:

Destination	Route	Distance (m)
1	-	0
2	1 - 2	160
3	1 - 2 - 3	560
4	1 - 2 - 5 - 4	630
5	1 - 2 - 5	510
6	1 - 2 - 5 - 6	580
7	1 - 2 - 5 - 6 - 7	620
8	1 - 2 - 5 - 6 - 7 - 8	770
9	1 - 2 - 5 - 6 - 7 - 8 - 9	940
10	1 - 2 - 5 - 6 - 7 - 8 - 9 - 10	1030
11	1 - 2 - 5 - 6 - 7 - 11	910

Full detail of algorithm:
Apply Djikstra's algorithm as follows.
Here is the distance information presented as a tab, which is a little easier to read.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0	160	-	-	-	590	-	-	-	-	-
2	160	0	400	-	350	-	-	-	-	-	-
3	-	400	0	280	-	-	-	-	-	-	-
4	-	-	280	0	120	-	-	190	-	-	-
5	-	-	-	120	0	70	-	-	-	-	-
6	590	-	-	-	70	0	40	-	-	-	-
7	-	-	-	-	-	40	0	150	-	-	290
8	-	-	-	190	-	-	150	0	170	260	-
9	-	-	-	-	-	-	-	170	0	220	110
10	-	-	-	-	-	-	-	260	220	0	350
11	-	-	-	-	-	-	290	-	110	350	0

- Initialise. $S = \{1\}$, $D = (0, \mathbf{160}, -, -, -, \mathbf{590}, -, -, -, -, -)$
- Iteration 1
  - Step 1. $S = \{1, 2\}$, $D = (0, 160, -, -, -, 590, -, -, -, -, -)$
  - Step 2. $S = \{1, 2\}$,

$$\begin{aligned} D &= (0, 160, 160 + 400, -, 160 + 350, 590, -, -, -, -, -) \\ &= (0, 160, \mathbf{560}, -, \mathbf{510}, 590, -, -, -, -, -) \end{aligned}$$

- Iteration 2
  - Step 1. $S = \{1, 2, 5\}$, $D = (0, 160, 560, -, 510, 590, -, -, -, -, -)$
  - Step 2. $S = \{1, 2, 5\}$,

$$\begin{aligned} D &= (0, 160, 560, 510 + 120, 510, \min(590, 510 + 70), -, -, -, -, -) \\ &= (0, 160, 560, \mathbf{630}, 510, \mathbf{580}, -, -, -, -, -) \end{aligned}$$

- Iteration 3
  - Step 1. $S = \{1, 2, 5, 3\}$, $D = (0, 160, 560, 630, 510, 580, -, -, -, -, -)$
  - Step 2. $S = \{1, 2, 5, 3\}$,

    $D = (0, \min(160, 560 + 400), 560, \min(630, 560 + 280), 510, 580, -, -, -, -, -)$
    $= (0, 160, 560, 630, 510, 580, -, -, -, -, -)$

- Iteration 4
  - Step 1. $S = \{1, 2, 5, 3, 6\}$, $D = (0, 160, 560, 630, 510, 580, -, -, -, -, -)$
  - Step 2. $S = \{1, 2, 5, 3, 6\}$,

    $D = (0, 160, 560, 630, \min(510, 580 + 70), 580, 580 + 40, -, -, -, -)$
    $D = (0, 160, 560, 630, 510, 580, \mathbf{620}, -, -, -, -)$

- Iteration 5
  - Step 1. $S = \{1, 2, 5, 3, 6, 7\}$, $(0, 160, 560, 630, 510, 580, 620, -, -, -, -)$
  - Step 2. $S = \{1, 2, 5, 3, 6, 7\}$,

    $D = (0, 160, 560, 630, 510, \min(580, 620 + 40), 620, 620 + 150, -, -, 620 + 290)$
    $D = (0, 160, 560, 630, 510, 580, 620, \mathbf{770}, -, -, \mathbf{910})$

- Iteration 6
  - Step 1. $S = \{1, 2, 5, 3, 6, 7, 4\}$,
    $D = (0, 160, 560, 630, 510, 580, 620, 770, -, -, 910)$
  - Step 2. $S = \{1, 2, 5, 3, 6, 7, 4\}$,

    $D = (0, 160, \min(560, 630 + 280), 630, \min(510, 630 + 120), 580, 620,$
    $\min(770, 630 + 190), -, -, 910)$
    $D = (0, 160, 560, 630, 510, 580, 620, 770, -, -, 910)$

- Iteration 7
  - Step 1. $S = \{1, 2, 5, 3, 6, 7, 4, 8\}$,
    $D = (0, 160, 560, 630, 510, 580, 620, 770, -, -, 910)$
  - Step 2. $S = \{1, 2, 5, 3, 6, 7, 4, 8\}$,

    $D = (0, 160, 560, \min(630, 770 + 190), 510, 580, \min(620, 770 + 150), 770,$
    $770 + 170, 770 + 260, 910)$
    $D = (0, 160, 560, 630, 510, 580, 620, 770, \mathbf{940}, \mathbf{1030}, 910)$

- Iteration 8
  - Step 1. $S = \{1, 2, 5, 3, 6, 7, 4, 8, 11\}$,
    $D = (0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$

- Step 2. $S = \{1, 2, 5, 3, 6, 7, 4, 8, 11\}$,

  $$D = (0, 160, 560, 630, 510, 580, \min(620, 910 + 290), 770, \min(940, 910 + 110), \min(1030, 910 + 350), 910)$$
  $$D = (0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$$

- Iteration 9
  - Step 1. $S = \{1, 2, 5, 3, 6, 7, 4, 8, 11, 9\}$,
    $D = (0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$
  - Step 2. $S = \{1, 2, 5, 3, 6, 7, 4, 8, 11, 9\}$,

    $$D = (0, 160, 560, 630, 510, 580, 620, \min(770, 940 + 170), 940, \min(1030, 940 + 220), \min(910, 940 + 110))$$
    $$D = (0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$$

- Iteration 10
  - Step 1. $S = \{1, 2, 5, 3, 6, 7, 4, 8, 11, 9, 10\}$
    $D = (0, 160, 560, 630, 510, 580, 620, 770, 940, 1030, 910)$
  - Stop.