Assignment 5: Solutions

TOTAL MARKS: 20

1. (a) 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	43	41	27	3	15	50
w_i	20	19	14	16	7	28
v_i/w_i	2.15	2.16	1.93	2	2.14	1.79

Since variable x_2 has the best value to weight (v_i/w_i) ratio and there is no integer (binary) requirement, we should make x_2 as large as possible i.e. $x_2^* = 45/19 = 2.3684$, and all other $x_i^* = 0$. Then $z^* = 41 \times x_2^* \approx 97.1053$.

[4 marks]

(b) The solution is $\mathbf{x}^T = (1, 1, 0, 0, 0.8571, 0)$ with $z^* = 96.8571$

[4 marks]

(c) The solution is $\mathbf{x}^T = (1, 0, 0, 1, 1, 0)$ with $z^* = 90$.
AMPL specification

Model file.

```
param n;
param w{i in 1..n};
param v{i in 1..n};
param max_weight;

var z{i in 1..n} >= 0 binary;
# var x{i in 1..n} >= 0, <= 1;
# var x{i in 1..n} >= 0 binary;

maximize value:    sum{i in 1..n} v[i]*z[i];
subject to weight: sum{i in 1..n} w[i]*z[i] <= max_weight;
```

Data file.

```
param n := 6;
param total_volume := 45;

param: v    w :=
  1    43  20
  2    41  19
  3    27  14
  4    32  16
  5    15   7
  6    50  28;
```

Comment out/in the appropriate variable line for integer or non-integer constraints.
NB: You don't need a separate data and model file to get the solution here, but it makes you code more adaptable and reusable.
NB: The question asked for AMPL – you get zero for Matlab code.

[4 marks]

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

[2 marks]

(e) Rounding the solution in (a) gives $\mathbf{x} = (0, 2, 0, 0, 0, 0)$ or $(0, 3, 0, 0, 0, 0)$ neither of which is feasible for the 0-1 (binary) (ILP). Rounding the solution in (b) gives $\mathbf{x} = (1, 1, 0, 0, 1, 0)$ or $(1, 1, 0, 0, 0, 0)$ both of which are feasible, but neither of which is optimal, for the 0-1 (binary) (ILP).

[2 marks]

(f) In the standard greedy approach, we sort the items in terms of their relative value, *i.e.,* , it an item has value c_i and volume a_i, then we give precedence to higher c_i/a_i. The we add them in this order until we run out of space. For the items in question the relative values are

$$(2.1500, 2.1579, 1.9286, 2.0000, 2.1429, 1.0714)$$

So we add items (in order) 2, and 1, and then the space remaining is too small for another item. The total value is then $z^* = 84$, which is worse than the optimal (as expected).

[2 marks]

(g) The computations required are one divide per item, and then we sort all of the items. The complexity of a mergesort in the worst case is $O(n \log n)$, which dominates over the divides (and also subsequent selection). So the overall complexity is $O(n \log n)$.

[2 marks]

Additional information: we could put the solution through MATLAB using

```
>> c=-[43;41;27;32;15;30];
>> a=[20 19,14,16,7,28];
>> b=45;
>> lb=zeros(size(c));
>> ub=ones(size(c));
```

We use $c = -[43; 41; 27; 32; 15; 30]$, because *linprog* minimises! Then we could use

```
[x,f] = linprog(c,a,b,[],[],lb)
[x,f] = linprog(c,a,b,[],[],lb,ub)
[x,f] = intlinprog(c,1:length(c),a,b,[],[],lb,ub)
```

to solve the LP and IP versions of the problem.