This document contains the key ideas to solve A1 questions. Note that many details may be ignored in several places.

Q2.
Assume the road starts at $a$, ends at $b$. There are $n$ houses at points $x_i$ where $a \leq x_i \leq b$. We represent each house by an interval $I_i$ that starts at $\max(a, x_i - 4)$ and $\min(b, x_i + 4)$. Identifying the minimum base stations that cover all houses is equivalent to maximizing the number of non-overlapping intervals (why?). Therefore the earliest finish time greedy algorithm (EFT) solves the problem. Place a base station at the end point of each selected interval.

Q3.
We need to remove 7 edges. In a graph with distinct edges, the edge with maximum weight in any cycle does not belong to the MST (why?). You can find a cycle using BFS algorithm. Thus one algorithm follows: Find a cycle by DFS ($O(n)$). Find the edge with maximum weight in that cycle ($O(n)$). Remove that edge. Continue this for 7 items. Thus the total run time is $O(n)$.

Q4.
The longest codeword (the maximum height of the huffman tree) is $n - 1$. Note that the huffman tree is a binary tree where each inner node has two children. So a huffman tree with a larger height is not possible.
Example: $f_1 = 1$, $f_2 = 1$, $f_3 = 2$, $f_4 = 4$, $f_5 = 8$, $f_6 = 16$, ···, $f_n = 2^{n-2}$. In general it happens when $f_i \geq \sum_{j=1}^{i-1} f_j$.

Q5.
Proof by contradiction. All of the numbers are in percentage.

![Huffman Tree Diagram]

Assume there exist one character ($z$) with a frequency $> 40$ but there is no codeword of length 1.
Note that the huffman algorithm sorts the characters according to the frequency (non-decreasing) and creates the tree based on this ordering. Assume character $z$ has a frequency $f_z > 40$. Since there is no codeword of length 1, in the last iteration two inner nodes are merged to create the root node $r$. WLOG, assume character $z$ is in the right subtree. Thus, the root of the right subtree is composed of a node with frequency greater than 40 and a node with frequency $f_x$. Moreover, assume the root of the left subtree has a frequency of $f_y$. Therefore, $f_x \leq f_y$ and $f_z \leq f_y$. Since $f_z > 40$, $f_y > 40$. Thus, $f_x < 20$ (sum of all frequencies is 100). Now, assume the children of the node with frequency $f_y$ have frequencies $f_{y_1}$ and $f_{y_2}$. Either (1) these two nodes are considered and merged before node $z$ then they should have a frequency of more than 40 that is not possible, or (2) these nodes are considered and merged before node $z$ so $f_{y_1} \leq f_x$ and $f_{y_2} \leq f_x$. Thus $f_y = f_{y_1} + f_{y_2} < 40$. This is in contradiction with $f_y > 40$. Hence the first assumption is incorrect.

Q6.
Assume $OPT$ is an optimal solution and $S$ is the result of EFT. We define a map $h : OPT \rightarrow S$. Similar to lecture, define $h(I) = J$ where $I \in OPT; J \in S; J$ conflicts $I$; and among all intervals in $S$ conflicting $I$, $J$ has the earliest finish time. Note that if $h(I) = J$ then $f(J) \leq f(I)$ otherwise $I$ should be in $S$ not $J$.

First $h$ is a function: $h(I)$ has a unique value. (why?)

Second, at most two jobs in $OPT$ can be mapped to a single job in $S$. For this part we use proof by contradiction technique. Assume there are three intervals $I_1, I_2, I_3 \in OPT$ that are mapped to the same interval $J \in S$. So all of these intervals should conflict with $J$ either by having the same type or overlapping. At most one of $I_1, I_2, and I_3$ can have the same type as $J$ otherwise $OPT$ is conflicting and not a solution. Thus, at least two intervals (say $I_1$ and $I_2$) overlap with $I$. Since $f(J) \leq f(I_1)$ and $f(J) \leq f(I_2)$: $f(J) \in I_1$ and $f(J) \in I_2$. Thus, $I_1$ and $I_2$ overlap and $OPT$ is conflicting and not a solution.

In all cases $OPT$ is not a solution. This is a contradiction. Hence EFT is a 2-approximation algorithm for JISP.

Q7.
Semantic array: $V[i]$ contains the maximum total weight of an independent set utilizing nodes 1 to $i$.

$V[0] = 0$.

$V[i] = \max(V[i - 1], w_i + V[i - 2])$. 


1 \( V[0] = 0 \)
2 \( \text{for } i = 1 \text{ to } n \text{ do} \)
3 \[ Used[i] = false \]
4 \( \text{for } i = 1 \text{ to } n \text{ do} \)
5 \[ V[i] = V[i - 1] \]
6 \( \text{if } V[i] < w_i + V[i - 2] \text{ then} \)
7 \[ V[i] = w_i + V[i - 2] \]
8 \[ Used[i] = true \]
9 \( S = \{ \} // \text{the optimal independent set} \)
10 \( i = n \)
11 \( \text{while } i > 0 \text{ do} \)
12 \( \text{if } Used[i] \text{ then} \)
13 \[ S = S \cup \{V_i\} \]
14 \( i = i - 2 \)
15 \( \text{else} \)
16 \[ i = i - 1 \]