Question 1 (Shortest Path, 30 points). Give the shortest path from S to T in the following graph (give the path):

We compute shortest path lengths from S using the algorithm for shortest paths in a DAG. The red numbers are the resulting values. We can then backtrack to find the path. In particular, for each vertex $v$ we find the proceeding vertex $w$ so that $d(v) = d(w) + \ell(w, v)$ and trace them backwards to find the final path of SABEGHIT as shown below:
Question 2 (Best Leaf, 35 points). You are given a balanced binary tree $T$ with $n$ leaves. Each leaf is labelled with a real number, and each other vertex is labelled with an operation either $+a$ or $\times b$ for $a$ or $b$ a positive real number. You want to pick a leaf and apply the relevant operations one at a time going up the tree. You would like to find the leaf that maximizes this value. For example in the example below, the selected leaf gives the largest value of $(5 \times 2) + 1 = 11$.

![Binary Tree Example](image)

*Give an algorithm to compute the largest possible value achievable this way. For full credit your algorithm should run in time $O(n)$ or better.*

We use divide an conquer. We note that the operations are all increasing in their inputs and so that largest possible outcome is obtained by getting the largest possible outcome at the previous level. We thus, define the following recursive function:

MaxValue($T$)
- If $T$ consists of only a single leaf
  - Return value of leaf
- Let $T_1$ and $T_2$ be the subtrees of $T$’s children
- Let $M = \text{Max}(\text{MaxValue}(T_1), \text{MaxValue}(T_2))$
- Let $P$ be the operation at $T$’s root
- Return $P(M)$

The runtime involves a constant amount of work plus two recursive calls on subtrees of half the size. Thus, the runtime is given by the recurrence $T(n) = 2T(n/2) + O(1)$. Thus, by the Master Theorem, the runtime is $O(n)$. 


Question 3 (Farm Work, 35 points). Lillian is a farmhand and is trying to plan her work for the season. She has a map (an undirected graph) of the country in which she works along with transportation costs for the various roads. She needs to plan work for both apple picking season and banana picking season. At each city (vertex) she knows the pay of the best apple picking job at that location and the pay of the best banana picking job. She wishes to plan a route from her home to an apple picking job, then to a banana picking job (possibly in the same city) and then back home in such a way as to maximize her income minus expenses. Give an algorithm to compute the best possible net profit. For full credit, your algorithm should run in near-linear time.

Our algorithm runs as follows: Let $G$ be the graph of the road network. Construct a new graph $G'$ consisting of three copies of $G$ ($G_0, G_1, G_2$) with extra edges between the corresponding vertices of $G_0$ and $G_1$ and the corresponding vertices of $G_1$ and $G_2$. Give $G'$ edge weights where an edge in one of the copies of $G$ is as usual. An edge from $G_0$ to $G_1$ is minus the pay from apple picking at that location and an edge from $G_1$ to $G_2$ has weight minus the payment for banana picking.

We claim that the answer is negative the weight of the shortest path in $G'$ from the copy of Lillian’s house in $G_0$ ($h_0$) to the copy in $G_2$ ($h_2$). This is because any such path travels to a city in $G_0$, switches to $G_1$, travels to some other city in $G_1$, switches to $G_2$ and returns home. The cost of this path is exactly equal to the cost of transit from Lillian’s home to the first city, then the second then back home minus the pay she would get for apply picking in the first and banana picking in the second.

Unfortunately, running Bellman-Ford on this graph is too time consuming. Instead, we compute $M_A$ and $M_B$ the maximum payout for picking apples and bananas, respectively. We then construct a new graph $G''$ from $G'$ by adding $M_A$ to all edges from $G_0$ to $G_1$ and $M_B$ to all edges from $G_1$ to $G_2$. This gives a graph with non-negative weights where the cost of a path in $G''$ is exactly $M_A + M_B$ more than the corresponding path in $G'$. Thus, the answer is $M_A + M_B$ minus the length of the shortest path from $h_0$ to $h_2$ in $G''$. This can be computed using Dijkstra’s algorithm.

The total runtime of this algorithm is $O(|V| \log |V| + |E|)$. We note that $G'$ (and thus $G''$) has only $3|V|$ vertices and $3|E| + 2|V|$ edges. It is easy to see that it is constructed in linear time. Running Dijkstra on it takes $O(|V| \log |V| + |E|)$ time and getting the final answer from there is constant time.