Solutions for Selection Contest 29 April

Question 1 Observe that each edge has exactly two end points. Hence, in the incidence matrix, there are exactly two 1's in a column. Consider the element in i-th row j-th column of $A^T A$. The value of this element is the dot product of the i-th column and the j-th column of A. The dot product is 2, if i and j are the same; it is 1, if the i-th edge and j-th edge share exactly one endpoint; it is 0, if the edges do not share any endpoints.

So, the desired sum is $2M + \sum_{i=1}^{N} \text{degree}(i) \cdot (\text{degree}(i) - 1)$. This can be calculated in linear time.

Question 2 This question essentially asks how many unordered pairs (i, j) satisfy $A_i + A_j \leq M$. The naive approach is to enumerate all the pairs and check the feasibility, which requires $O(n^2)$ time and cannot be accepted.

The intended solution for this question is $O(n \log(n))$. We start by sorting A_i in increasing order, which can be done in $O(n \log(n))$ time. For a fixed i $(1 \leq i \leq n)$, we consider the largest j, such that $j > i$ and $A_i + A_j \leq M$. It is easy to see that for this fixed i, all the k that satisfy $k > i$ and $A_i + A_k \leq M$ are exactly lying in the interval $(i, j]$. Therefore, i contributes $j - i$ pairs to the answer. Sum over all the i's to get the result. It remains to calculate j for fixed i, and this can be done by using binary search, which takes $O(\log(n))$ time.

There is actually a linear time algorithm to find j for fixed i. If you are interested, you can look at the referenced program. This implies that if the input is sorted, then this question can be solved in linear time.

Question 3 The suggested approach for this question is Segment Tree. This is essentially the use of dual variable, which we have discussed in the second lecture.

The idea is to maintain a segment tree with "min" and "max" fields simultaneously. When negating, swap and negate "min" and "max".

Similar questions can be found in our Segment Tree excercises.

Question 4 An easy observation is that there is no point to switch a light twice. The naive approach is to enumerate which lights to switch, but this will take $O(2^{mn})$ time, which is ridiculously large.

Since our goal is to turn all the lights on, the key observation is that if we fix the operations for the first row (here operations refer to the way of switching the lights), then all the operations for all the other lights are fixed, in order to make sure all the lights are turned on. Therefore, the algorithm is to enumerate the first row, which takes $O(2^m)$, and verify the others, which takes $O(mn)$.

But $2^m \cdot mn$ is still a bit large for this problem. Since m is at most 16, we can encode the operation for a single row in a binary number at most 2^{16}, and we can do some precomputation to further decrease the verifying cost. At last, we can achieve $O(2^m \cdot n)$ complexity. For details, you can have a look at our referenced program.