**SHEET- 15**

**Greedy Algorithm Approach**

1. Given an array of jobs with different time requirements. There are K identical assignees available and we are also given how much time an assignee takes to do one unit of the job. Find the minimum time to finish all jobs with following constraints.

* An assignee can be assigned only contiguous jobs. For example, an assignee cannot be assigned jobs 1 and 3, but not 2.
* Two assignees cannot share (or co-assigned) a job, i.e., a job cannot be partially assigned to one assignee and partially to other.

Input:

K: Number of assignees available.

T: Time taken by an assignee to finish one unit of job

job[]: An array that represents time requirements of different jobs.  
  
Example:

Input: k = 2, T = 5, job[] = {4, 5, 10}

Output: 50

The minimum time required to finish all the jobs is 50.

There are 2 assignees available. We get this time by assigning {4, 5} to first assignee and {10} to second assignee.

Input: k = 4, T = 5, job[] = {10, 7, 8, 12, 6, 8}

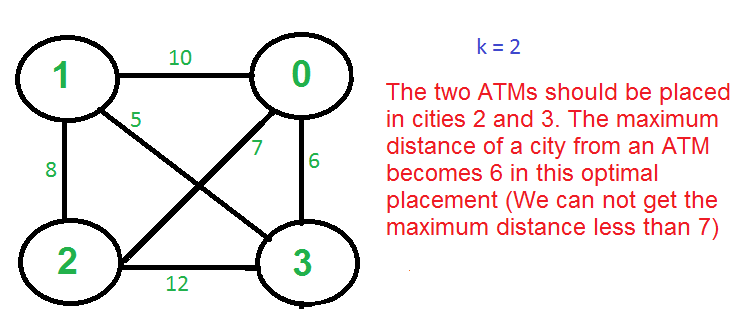
Output: 75

We get this time by assigning {10} {7, 8} {12} and {6, 8}

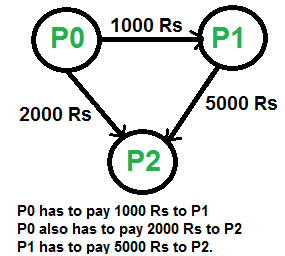
1. K Centers Problem

Given n cities and distances between every pair of cities, select k cities to place warehouses (or ATMs or Cloud Server) such that the maximum distance of a city to a warehouse (or ATM or Cloud Server) is minimized.

For example consider the following four cities, 0, 1, 2, and 3, and distances between them, how to do place 2 ATMs among these 4 cities so that the maximum distance of a city to an ATM is minimized.



1. Given a number of friends who have to give or take some amount of money from one another. Design an algorithm by which the total cash flow among all the friends is minimized.



1. Operations on a tree

You are given an undirected tree G with N nodes. You are also given an array A of N integer elements where A[i] represents the value assigned to node i and an integer K.

You can apply the given operation on the tree at most once:

* Select a node x in the tree and consider it as the root of the tree.
* Select a node y in the tree and update the value of each node in the subtree of y by taking its XOR with K. That is, for each node u in the subtree of node y, set A[u] = A[u] XOR K.

Find the maximum sum of values of nodes that are available in the tree, after the above operation is used optimally.

**Note**

* Assume 1-based indexing.
* XOR represents the bitwise XOR operator.

**Input format**

* The first line contains a single integer T that denotes the number of test cases.
* For each test case:
  + The first line contains an integer N.
  + The second line contains an integer K.
  + The third line contains N space-separated integers denoting array A.
  + Next N−1 lines contain two space-separated integers denoting an edge between node u and v.

**Output format**

For each test case, print the maximum possible sum of values of nodes present in the tree. Print the output for each test case in a new line.

**Sample Input**

2

3

4

4 4 4

1 2

1 3

4

2

5 1 4 2

1 2

2 3

3 4

**Sample Output**

12

18

**Explanation**

**For test case 1:**

* If the operation is applied on the tree, then the sum of node values will decrease. Thus, it is optimal to not apply operation on the tree.
* Maximum possible sum of node values is *A[1] + A[2] + A[3] = 4 + 4 + 4 = 12.*
* Thus, output *12*.

For test case 2:

* Consider *x = 4,*for the operation. This means the tree is rooted at node 4*.*
* Consider *y = 3,*for the operation. This means we need to update the values of nodes present in the subtree of node *3* i.e. nodes *1, 2, and 3*.
  + *A[3] = A[3] XOR K = 4 XOR 2 = 6*
  + *A[2] = A[2] XOR K = 1 XOR 2 = 3*
  + *A[1] = A[1] XOR K = 5 XOR 2 = 7*
* Sum of values of all the nodes is *A[1] + A[2] + A[3] + A[4] = 6 + 3 + 7 + 2 = 18,*which is maximum possible sum.
* Thus, output *18.*

1. Minimize the nodes

You are given N strings. Each string is given in the form of an array *cnt* of size *26* where the ith element in array denotes the count of the ith character of lowercase English alphabets in the string.

You have to select exactly *4* strings and insert the strings into a *Trie*. You are allowed to shuffle the characters in each string.

**Task**

Determine the minimum number of nodes present in *Trie* if the strings are selected optimally.

**Notes**

* *Trie*will always have a root node. Include root node in the count of nodes present in Trie.
* Insertion proceeds by walking the Trie according to the string to be inserted, then appending new nodes for the suffix of the string that is not contained in the Trie.

**Input format**

* The first line contains a single integer T which denotes the number of test cases.
* The first line of each test case contains an integer N.
* The next N lines of each test case contain *26* space-separated integers denoting the count of each lowercase English alphabet in the string.

**Output format**

For each test case, print the minimum number of nodes present in Trie in a new line.

**Sample Input**

1

5

2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

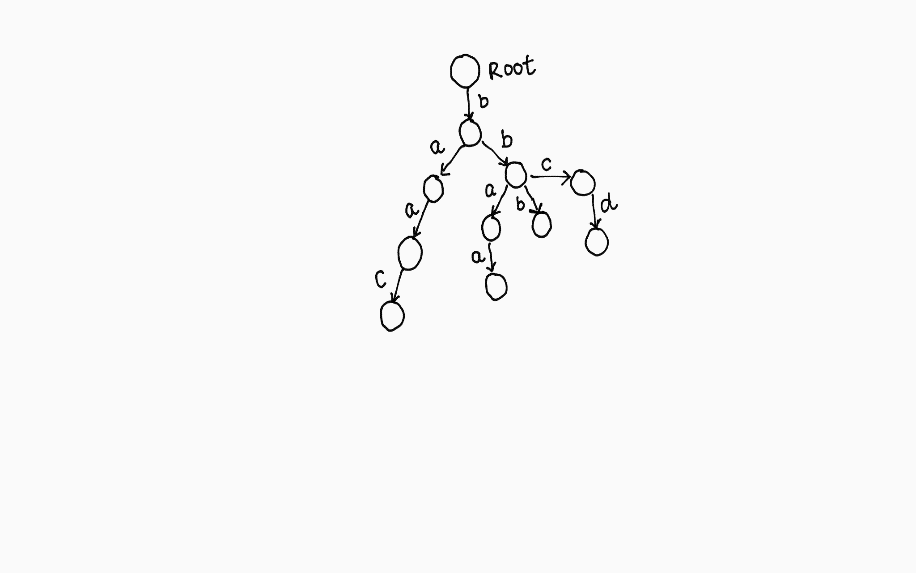
**Sample Output**

10

**Explanation**

*Approach*

* You can shuffle the characters of *5* strings such that strings are:
  + *baac*
  + *bbcd*
  + *bbb*
  + *bbaa*
  + *bccc*
* Now, there can be *5*different possibilities to select *4*strings.
* If you select the first *4*strings:
  + *baac, bbcd, bbb, bbaa:* Minimum number of nodes in Trie will be *10* (including root node). The Trie will look like:



* It can be proved that you cannot make a Trie with less than *10*nodes.
* Thus, required answer is *10*.