General Ideas

md.mottakin.chowdhury

November 2018

1 GCD on subsegments

Assume you have set of numbers in which you add elements one by one and on each step calculate *gcd* of all numbers from set. Then we will have no more than $log(a[i])$ different values of *gcd*. Thus you can keep compressed info about all *gcd* on Subsegments of $a[i]$:

```
    int a[n];
    map<int, int> sub_gcd[n];
    /*
    Key is gcd,
    Value is the largest length such that gcd(a[i - len], ..., a[i]) equals to key.
    */
    sub_gcd[0][a[0]] = 0;
    for(int i = 1; i < n; i++)
    {
        sub_gcd[i][a[i]] = 0;
        for(auto it: sub_gcd[i - 1])
        {
            int new_gcd = __gcd(it.first, a[i]);
            sub_gcd[i][new_gcd] = max(sub_gcd[i][new_gcd], it.second + 1);
        }
}
```

2 From Static Set to Expandable via $O(log(n))$

Assume you have some static set and you can calculate some function f of the whole set such that $f(x_1, ..., x_n) = g(f(x_1, ..., x_{k-1}), f(x_k, ..., x_n))$, where g is some function which can be calculated fast. For example, $f()$ as the number of elements less than k and $g(a + b) = a + b$. Or $f(S)$ as the number of occurences of strings from S into T and g is a sum again.

With additional $log(n)$ factor you can also insert elements into your set. For this let's keep $log(n)$ disjoint sets such that their union is the whole set. Let the size of k^{th} be either 0 or 2^k depending on binary presentation of the whole set size. Now when inserting element you should add it to 0^{th} set and rebuild every set keeping said constraint. Thus k^{th} set will tell $F(2^k)$ operations each 2^k steps where $F(n)$ is the cost of building set over n elements from scratch which is usually something about n.

3 XOR Subsets

Assume you have set of numbers and you have to calculate something considering xors of its subsets. Then you can assume numbers to be vectors in k-dimensional space over field Z_2 of residues modulo 2. This interpretation useful because ordinary methods of linear algebra work here. For example, here you can see how using gaussian elimination to keep basis in such space and answer queries of k^{th} largest subset xor:

```
    // Problem: in each query, either add a number to the set, or find the k-th
    // largest subset xor from the set. So if the set is currently {1,2} then subset // xors are {0,1,2,3
```

```
int b[32];
int sz = 0;

void add(int x)
{
    for(int i = 0; i < sz; i++)
        if((x ^ b[i]) < x)
            x ^= b[i];
    for(int i = 0; i < sz; i++)
        if((x ^ b[i]) < b[i])
            b[i] ^= x;
    if(x)
    {
        b[sz++] = x;
        for(int i = sz - 1; i; i--)
            if(b[i] < b[i - 1])
                swap(b[i], b[i - 1]);
    }
}
int get(int k)
{
    k--;
    int ans = 0;
    for(int i = 0; i < sz; i++)
        if((k >> i) & 1)
            ans ^= b[i];
    return ans;
}
int main()
{
    int n;
    cin >> n;
    while(n--)
    {
        int t, x;
        cin >> t >> x;
        if(t == 1)
            add(x);
        else
            cout << get(x) << "\n";
    }
}
```

4 Cycles in Graph as Linear Space

Assume every set of cycles in graph to be vector in E-dimensional space over Z_2 having one if corresponding edge is taken into set or zero otherwise. One can consider combination of such sets of cycles as sum of vectors in such space. Then you can see that basis of such space will be included in the set of cycles which you can get by adding to the tree of depth first search exactly one edge. You can consider combination of cycles as the one whole cycle which goes through 1-edges odd number of times and even number of times through 0-edges. Thus you can represent any cycle as combination of simple cycles and any path as combination as one simple path and set of simple cycles. It could be useful if we consider pathes in such a way that going through some edge twice annihilates its contribution into some final value. Example: find path from vertex u to v with minimum xor-sum.

5 Matrix Exponentiation Optimization

Assume we have $n \times n$ matrix A and we have to compute $b = A^m x$ several times for different m. Naive solution would consume $O(qn^3 log(n)$ time. But we can precalculate binary powers of A and use $O(log(n))$ multiplications of matrix and vector instead of matrix and matrix. Then the solution will be $O((n^3 + qn^2)log(n)$.

6 Euler Tour Technique

You have a tree and there are lots of queries of kind add number on subtree of some vertex or calculate sum on the path between some vertices.

Let's consider two euler tours: in first we write the vertex when we enter it, in second we write it when we exit from it. We can see that difference between prefixes including subtree of v from first and second tours will exactly form vertices from v to the root. Thus problem is reduced to adding number on segment and calculating sum on prefixes.

7 From Expandable Set to Dynamic via $O(log(n))$

Assume for some set we can make non-amortized insert and calculate some queries. Then with additional $O(log(n))$ factor we can handle erase queries. Let's for each element x find the moment when it's erased from set. Thus for each element we will wind segment of time $[a, b]$ such that element is present in the set during this whole segment. Now we can come up with recursive procedure which handles $[l, r]$ time segment considering that all elements such that $[l, r] \subset [a, b]$ are already included into the set. Now, keeping this invariant we recursively go into $[l, m]$ and $[m, r]$ subsegments. Finally when we come into segment of length 1 we can handle the query having static set.