

1 全体のテンプレ

別にこれに固定しなくてもよい

```
1  /*
2  input data
3  */
4
5  void init(){
6  }
7
8  bool input(){
9  }
10
11
12  /*
13  dp とか
14  */
15  void init_solve(){
16  }
17
18  void q_input(){
19  }
20
21  int solve(){
22      return 0;
23  }
24
25
26  int main(){
27      int m;
28      for(int i=0;i<m;i++){
29          cout<<solve()<<endl;
30      }
31  }
```

2 include

```
1  #include<cmath>
2  #include<string>
3  #include<vector>
4  #include<queue>
5  #include<stack>
6  #include<set>
7  #include<map>
8  #include<algorithm>
9  #include<utility>
10 #include<climits>
11 #include<cstdio>
12 #include<cassert>
13 using namespace std;
```

3 素数

```
1 vector<int> prime;
2 vector<int> prime_list;
3 void prime_set(int n){
4     n+=100;
5     prime.resize(n);
6     for(int i=0;i<n;i++){
7         prime[i]=1;
8     }
9     prime[0]=prime[1]=0;
10    for(int i=0;i<n;i++){
11        if(prime[i]){
12            for(int j=i*2;j<n;j+=i){
13                prime[j]=0;
14            }
15        }
16    }
17    for(int i=0;i<n;i++){
18        if(prime[i])prime_list.push_back(i);
19    }
20 }
```

4 文字列

```
1 string revStr(string s){
2     return string(s.rbegin(),s.rend());
3 }
```

5 二次元幾何

```
1 #define X first
2 #define Y second
3
4 typedef long double ld;
5 typedef long long ll;
6 typedef ll P_type;
7 typedef pair<ll,ll> P;
8
9 const ld INF = 1e39;
10 const ld EPS = 1e-8;
11 const ld PI = acos(-1);
12
13 P_type out_pro(P a,P b) {
14     return (a.X * b.Y - b.X * a.Y);
15 }
16
17 P_type in_pro(P a,P b){
18     return (a.X * b.X + a.Y * b.Y);
19 }
20
21 //以下2つは小数点以下が必要なことがあるのでにする ld
22 ld pow_len(P a) {
23     return a.X * a.X + a.Y * a.Y;
24 }
25
26 ld len(P a){
27     return sqrt(pow_len(a));
28 }
29
30 P make_v(P a,P b) {
31     return P(a.X - b.X, a.Y - b.Y);
32 }
33 //凸法
34 vector<P> convex(vector<P> list) {
35     int m=0;
36     vector<P> res(list.size()*2);
37     sort(list.begin(),list.end());
38     for(int i=0; i<list.size(); res[m++]=list[i]){
39         for(;m>1&&out_pro(make_v(res[m-1],res[m-2]),make_v(list[i],res[m-2]))<=EPS;--m);
40     }
41     for(int i=list.size()-2,r=m; i>=0; res[m++]=list[i--]){
42         for(;m>r&&out_pro(make_v(res[m-1],res[m-2]),make_v(list[i],res[m-2]))<=EPS;--m);
43     }
44     res.resize(m-1);
45     return res;
46 }
```

6 UnionFind

```
1 class union_find{
2 private:
```

```

3     vector<int> parents;
4     vector<int> rank;
5 public:
6     union_find(int n){
7         parents.resize(n);
8         rank.resize(n);
9     }
10    void init(){
11        for(int i=0;i<parents.size();i++){
12            parents[i]=i;
13            rank[i]=0;
14        }
15    }
16    int find(int x){
17        if(parents[x]==x){
18            return x;
19        }else{
20            return parents[x]=find(parents[x]);
21        }
22    }
23    void unite(int x,int y){
24        x=find(x);
25        y=find(y);
26        if(x==y)return;
27        if(rank[x]<rank[y]){
28            parents[x]=y;
29        }else{
30            parents[y]=x;
31            if(rank[x]==rank[y])rank[x]++;
32        }
33    }
34    bool same(int x,int y){
35        return (find(x)==find(y));
36    }
37 };

```