<Yourein/Set>

aset

任意のint型もしくはlong long型の整数集合(std::set)A, Bにおいて、$A \wedge B$を返します。

Difinition

```
std::set<long long> aset(std::set<long long> *s1, std::set<long long> *s2)
std::set<int> aset(std::set<int> *s1, std::set<int> *s2)
```

使用例

```
set<long long> a;
set<long long> b;

for (int i = 0; i <= 51; i++){
    if (i%5 == 0) a.insert(i); //aは51以下の5の倍数
}
for (int i = 0; i <= 31; i++){
    if (i%2 == 0) b.insert(i); //bは31以下の2の倍数
}

auto na = Yourein::sets::aset(&a, &b); //集合A, Bの論理積を取る

//集合A, Bの要素を出力
for (auto x : a){
    cout << x << ' ';
}
cout << endl;

for (auto x : b){
    cout << x << ' ';
}
cout << endl;

//論理積集合の要素を出力
for (auto x : na){
    cout << x << ' ';
}
cout << endl;
```

出力結果

```
0 5 10 15 20 25 30 35 40 45 50
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

```
0 10 20 30
```

注意点 関数に渡すsetはsetのポインタである必要があります。 また、int型、long long型以外のオーバーロードを持っていません。 それ以外の型のsetを渡そうとすると、コンパイルエラーになります。

計算量は$O(min(n(A),n(B)) \times log(max(n(A),n(B))))$ です。 ただし、$n(x)$を集合xの要素数と定めます。

nset

任意のint型もしくはlong long型の整数集合(std::set) S において、任意の開区間 $[L,R]$ 中の$\overline{S}$を返します。

Difinition

```
std::set<long long> nset(std::set<long long> *baseset, long long rangeL, long long rangeR)
std::set<int> nset(std::set<int> *baseset, int rangeL, int rangeR)
```

使用例

```
int main(){
    set<long long> a;

    //Make a Set for odd number
    for (int i = 0; i <= 51; i++){
        if (i%2 == 0) a.insert(i);
    }
    //Print S
    for (auto x : a){
        cout << x << " ";
    }
    cout << endl;

    //Calc NOT A in [12, 30]
    auto na = Yourein::sets::nset(&a, 12LL, 30LL);

    //Check NOT A in [12, 30]
    for (auto x : na){
        cout << x << ' ';
    }
    cout << endl;
}
```

出力結果

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
13 15 17 19 21 23 25 27 29
```

rangeLとrangeRが省略された場合は$[S_{min}, S_{max}]$で演算を行います。

```
auto na = Yourein::sets::nset(&a);

for (auto x : na){
    cout << x << " ";
}
cout << endl;
```

とすると、

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

となり、計算区間は$[S_{min}, S_{max}]$となります。

注意 この関数の第一引数にわたすsetはポインタである必要があります。 また、set::vector配列で表現された集合を扱うことはできません。予めstd::set型になおしてから使用してください。
計算量は$O([L,R] \times log([S_{min}, S_{max}])) = O((R-L) \times log(S_{max} - S_{min}))$ です。

計算量に関しては一切の改善をしていないので $R - L + 1 \leq 10^6$ であるように値を設定してください。 **関数内部の方もfast intではないので本当に計算が遅いです。**

getsubset

ある集合Aについて、そのAの補集合を空集合も含めてすべて列挙します。 ただし、使用しているアルゴリズムの関係上$n(A) \leq 20$の制限があります。

Difiniton

```
template<typename T> std::vector<std::vector<T>> getsubset(std::vector<T> basevec)
template<typename T> std::vector<std::vector<T>> getsubset(std::string basevec)
```

上の定義に従って、型<T>のvector配列を受け取り、補集合をvector化して配列とした二次元vector(vector<vector<T>>)を返します。

使用例

```
vector<double> a = {1, 2, 3, 4, 5};
```

```
auto v = Yourein::sets::getsubset<double>(a);

for (int i = 0; i < v.size(); i++){
    for (int j = 0; j < v[i].size(); j++){
        cout << v[i][j] << " ";
    }
    cout << endl;
}
```

出力結果

```
1
2
1 2
3
1 3
2 3
1 2 3
4
1 4
2 4
1 2 4
3 4
1 3 4
2 3 4
1 2 3 4
5
1 5
2 5
1 2 5
3 5
1 3 5
2 3 5
1 2 3 5
4 5
1 4 5
2 4 5
1 2 4 5
3 4 5
1 3 4 5
2 3 4 5
1 2 3 4 5
```

(先頭の空行は空集合を表す)

ただし、**string型を引数として投げるときはchar型をテンプレートに指定してください。**

```
string s = "Hello";

auto v = Yourein::sets::getsubset<char>(s);

for (int i = 0; i < v.size(); i++){
    for (int j = 0; j < v[i].size(); j++){
        cout << v[i][j] << " ";
    }
    cout << endl;
}
```

```
H
e
H e
l
H l
e l
H e l
l
H l
e l
H e l
l l
H l l
e l l
H e l l
o
H o
e o
H e o
l o
H l o
e l o
H e l o
l o
H l o
e l o
H e l o
l l o
H l l o
e l l o
H e l l o
```

注意点 $O(2^N)$になります。**めちゃくちゃオーダーが重いです。** わかる人向けに書くと、bit全探索をしています。

$n(A) \leq 20$程度であれば常識的な計算時間で全探索ができますが、それ以上になると探索に相当な時間を要します。 具体的に例を示すと以下の$O(2^{20})$プログラムの関数部分の計算時間は1711msです。 $O(2^N)$から、Nが増えると計算量も指数関数的に増えていくことがわかるので、$N > 20$はかなり長時間の計算が必要であることがわかります。

```
std::chrono::system_clock::time_point  start, end;
vector<int> a(20);
for (int i = 0; i < 20; i++){
    a[i] = i;
}

start = std::chrono::system_clock::now();

auto v = Yourein::sets::getsubset<int>(a);

end = std::chrono::system_clock::now();

cout << "Calculation time = " <<
std::chrono::duration_cast<std::chrono::milliseconds>(end-start).count() << endl;
```

そのため、この関数の使用時には渡す配列の要素数が20を超えないかをチェックしてから使うようにしてください。 この関数内部で要素数によって処理を打ち切るようなことは一切行っていません。

また、関数使用時と元配列宣言時の<>の中身(今回はint)は必ず一致している必要があります。

```
vector<int> a;
Yourein::sets::getsubset<int>(a);
```

仮にfloat, double型のvector配列をgetsubset関数に投げたいときは、

```
vector<float> b;
vector<double> c;

Yourein::sets::getsubset<float>(b);
Yourein::sets::getsubset<double>(c);
```

と、しっかり型名を意識して書いてください。