The Ultimate ICPC Torpedium

Jorge Fiestas

October 23, 2019

1 String Matching

1.1 KMP Algorithm

```
void border(string &s) {
  int n = s.size();
  b[0] = -1;

  for (int i = 1; i <= n; ++i) {
    b[i] = b[i-1];
    while (b[i] != -1 and s[i-1] != s[b[i]])
      b[i] = b[b[i]];
    b[i] += 1;
  }

  return b;
}
```

2 Tree Like Structures

2.1 Fenwick Tree (BIT)

```
void add(int *bit, int idx, int v) {
    while (idx <= n) {
        bit[idx] += v;
        idx += idx & (-idx);
    }
}

int sum(int *bit, int idx){
    int s = 0;
    while(idx > 0){
        s += bit[idx];
        idx -= idx & (-idx);
    }
    return s;
}
```

3 Miscelaneous

3.1 Fast Fourier Transform

```
using cd = complex<double>;
const double PI = acos(-1);

void fft(vector<cd> & a, bool invert) {
    int n = a.size();

    for (int i = 1, j = 0; i < n; i++) {
        int bit = n >> 1;
        for (; j & bit; bit >>= 1)
            j ^= bit;
        j ^= bit;

        if (i < j)
            swap(a[i], a[j]);
    }

    for (int len = 2; len <= n; len <<= 1) {
        double ang = 2 * PI / len * (invert ? -1 : 1);
        cd wlen(cos(ang), sin(ang));
        for (int i = 0; i < n; i += len) {
            cd w(1);
            for (int j = 0; j < len / 2; j++) {
                cd u = a[i+j], v = a[i+j+len/2] * w;
                a[i+j] = u + v;
                a[i+j+len/2] = u - v;
                w *= wlen;
            }
        }
    }

    if (invert) {
        for (cd & x : a)
            x /= n;
    }
}
```