Aerial Ace

Team Reference Material

Supplement for 2.0

1 SPFA 费用流

```cpp
struct Edge {
    Edge *next, *rev;
    int from, to, cap, cost;
} *last[maxn], *prev[maxn], e[maxm], *ecnt = e;
inline void link(int a, int b, int w, int c)
{
    *++ecnt = (Edge) {last[a], ecnt + 1, a, b, w, c}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], ecnt - 1, b, a, 0, -c}; last[b] = ecnt;
}
int s, t, q[maxn << 2], dis[maxn];
ll ans;
bool inq[maxn];
#define inf 0x7fffffff
inline bool spfa()
{
    for (int i = 1; i <= t; ++i) dis[i] = inf;
    int head = 0, tail = 1; dis[q[1] = s] = 0;
    while (head < tail)
    {
        int now = q[++head]; inq[now] = 0;
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (iter -> cap && dis[iter -> to] > dis[now] + iter -> cost)
            {
                dis[iter -> to] = dis[now] + iter -> cost;
                prev[iter -> to] = iter;
                !inq[iter -> to] ? inq[q[++tail] = iter -> to] = 1 : 0;
            }
    }
    return dis[t] != inf;
}
inline void mcmf()
{
    int x = inf;
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from]) cmin(x, iter -> cap);
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from])
    {
        iter -> cap -= x;
        iter -> rev -> cap += x;
        ans += 1ll * x * iter -> cost;
    }
}
```

2 广义后缀自动机

```cpp
struct sam {
    sam *next[26], *fa;
    int val;
} mem[maxn << 1], *tot = mem;
inline sam *extend(sam *p, int c)
{
    if (p -> next[c])
    {
        sam *q = p -> next[c];
        if (q -> val == p -> val + 1)
            return q;
        else
        {
            sam *nq = ++tot;
            memcpy(nq -> next, q -> next, sizeof nq -> next);
            nq -> val = p -> val + 1;
            nq -> fa = q -> fa;
            q -> fa = nq;
            for ( ; p && p -> next[c] == q; p = p -> fa)
                p -> next[c] = nq;
            return nq;
        }
    }
    sam *np = ++tot;
    np -> val = p -> val + 1;
    for ( ; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
    if (!p)
        np -> fa = mem;
    else
    {
        sam *q = p -> next[c];
        if (q -> val == p -> val + 1)
            np -> fa = q;
        else
        {
            sam *nq = ++tot;
            memcpy(nq -> next, q -> next, sizeof nq -> next);
            nq -> val = p -> val + 1;
            nq -> fa = q -> fa;
            q -> fa = np -> fa = nq;
            for ( ; p && p -> next[c] == q; p = p -> fa)
                p -> next[c] = nq;
        }
    }
    return np;
}
```