DP

Created	@2025년 5월 18일 오후 7:40
Tags	Bitset CHT D&C DP Knuth LIS SoS DP Tree DP 벌레캠프

Longest Increasing Subsequence
Convex Hull Trick
Divide and Conquer Optimization
Tree DP
Knuth Optimization
Subset of Sum DP
Bitset Optimization
Berlekamp-Massey

Longest Increasing Subsequence

BOJ 12015 - 가장 긴 증가하는 부분 수열 2

```
// Usage: vl result = lis(arr);
// Time Complexity: O(n*logn)
vl lis(vl &arr) {
    ll n=arr.size();
    vl tmp, from;
    for (ll x:arr) {
        ll loc=lower_bound(tmp.begin(), tmp.end(), x)-tmp.begin();
        if (loc==tmp.size()) tmp.push_back(x);
        else tmp[loc]=x;
        from.push_back(loc);
    }
    vl ret=vl(tmp.size());
    ll target=tmp.size()-1;
    for (ll i=n-1;i>=0;i--) {
        if (target==from[i])
```

```
            ret[target--]=arr[i];
    }
    return ret;
}
```

Convex Hull Trick

BOJ 13263 - 나무 자르기

```
// Usage: CHT cht; cht.addLine(b[0], D[0]);
//        for(i = 1; i < n; ++i) { D[i] = cht.query(a[i]); cht.addLine(b[i], D[i]); }
// Memo: O(n^2) → O(n*logn), 아래 조건 중 하나 만족해야 함
// Memo: D[i] = max[j<i](D[j]+b[j]*a[i]), (b[k]<=b[k+1])
// Memo: D[i] = min[j<i](D[j]+b[j]*a[i]), (b[k]>=b[k+1])
struct CHT {
    struct Line {
        ll m, b;   // y = m*x + b
    };
    vector<Line> lines;
    vl xs;
    ll intersect(Line &l1, Line &l2) {
        ll num = l1.b - l2.b;
        ll den = l2.m - l1.m;
        // min인 경우 num과 den에 -1을 곱함
        return num>=0 ? (num+den-1)/den : num/den;
    }
    void addLine(ll m, ll b) {
        if (lines.size() && lines.back().m==m) {
            if (lines.back().b>=b) return; // min인 경우 부등호 반대로
            lines.pop_back();
            xs.pop_back();
        }
        Line L{m,b};
        while (lines.size()) {
            ll x=intersect(lines.back(), L);
            if (x<=xs.back()) {
                lines.pop_back();
                xs.pop_back();
```

```
            }
            else break;
        }
        if (lines.empty()) {
            lines.push_back(L);
            xs.push_back(LLONG_MIN);
        }
        else {
            lines.push_back(L);
            xs.push_back(intersect(lines[lines.size()-2], L));
        }
    }
    ll query(ll x) {
        ll idx=upper_bound(xs.begin(), xs.end(), x)-xs.begin()-1;
        return lines[idx].m*x+lines[idx].b;
    }
};
```

Divide and Conquer Optimization

BOJ 13261 - 탈옥

```
// Usage: f(x, 0, n-1, 0, n-1) → D[x][i]가 모두 채워짐
// Memo: D[t][i] = min[j<i](D[t-1][j] + C[j][i])
// Memo: O(kn^2) → O(knlogn), 아래 조건 만족해야 함
// Memo: C[a][c] + C[b][d] <= C[a][d] + C[b][c], (a<=b<=c<=d)

void f(ll t, ll s, ll e, ll l, ll r){
    if(s>e) return;
    ll m=(s+e)>>1;
    ll opt=l;
    for(ll i=l;i<=r;i++){
        if(D[t-1][opt]+C[opt][m]>D[t-1][i]+C[i][m]) opt=i;
    }
    D[t][m]=D[t-1][opt]+C[opt][m];
    f(t, s, m-1, l, opt);
    f(t, m+1, e, opt, r);
}
```

Tree DP

```
// Tree DP
// Usage: dfs(dfs, root, -1);
// Note: 0-based
vvl adj(v);
vl dp(v);
auto dfs=[&adj, &dp](auto self, ll node, ll pa) {
    for (auto &e: adj[node]) {
        if (e==pa) continue;
        self(self, e, node);
        /* dp expression */
    }
};
```

Knuth Optimization

BOJ 11066 - 파일 합치기

```
// Note: O(n^3) → O(n^2), 아래 3가지 조건 만족
// D[i][j] = min{i<k<j}(D[i][k]+D[k][j]) + C[i][j]
// C[a][c]+C[b][d] <= C[a][d]+C[b][c], (a<=b<=c<=d)
// C[b][c] <= C[a][d]
// Usage: vvl dp = knuth(arr);
vvl knuth(vl &arr) {
    constexpr ll INF = 2e18;
    ll i, k;
    ll n=arr.size();
    vl s(n+1); // 누적 합 배열
    for (i=0;i<n;i++) s[i+1]=s[i]+arr[i];

    vvl dp(n+1, vl(n+1)), opt(n+1, vl(n+1));
    for (i=1;i<=n;i++) {
        dp[i-1][i]=0;
        opt[i-1][i]=i;
    }

    for (i=2;i<=n;i++) {
```

```
        for (ll l=0;i+l<=n;l++) {
            ll r=i+l;
            dp[l][r]=INF;
            ll start=opt[l][r-1];
            ll end=opt[l+1][r];
            for (k=start;k<=end;k++) {
                ll cost=dp[l][k]+dp[k][r]+s[r]-s[l];
                if (cost<dp[l][r]) {
                    dp[l][r]=cost;
                    opt[l][r]=k;
                }
            }
        }
    }

    return dp;
}
```

Subset of Sum DP

```
// Usage: vl F = sos_dp(n, A);
// Note: mask의 부분 집합 x에 대하여, F[mask] = sum of A[x]
// Note: A.size == 2^n
// time: O(N*2^N), memory: O(2^N)
vl sos_dp(ll n, vl &A) {
    ll i;
    vl F(1<<n);
    for (i=0;i<(1<<n);i++) F[i]=A[i];
    for (i=0;i<n;i++) for (ll mask=0;mask<(1<<n);mask++) {
        if (mask&(1<<i)) F[mask]+=F[mask^(1<<i)];
    }
    return F;
}
```

Bitset Optimization

```
// ==================== 이 부분은 항상 코드의 맨 윗부분에 있어야 함==========
#define private public
#include <bitset>
#undef private
#include <x86intrin.h>
// ================================================================
template <size_t _Nw>
void _M_do_sub(_Base_bitset<_Nw> &A, const _Base_bitset<_Nw> &B) {
    for (int i=0, c=0;i<_Nw;i++)
        c=_subborrow_u64(c, A._M_w[i], B._M_w[i], (ull*)&A._M_w[i]);
}
template <>
void _M_do_sub(_Base_bitset<1> &A, const _Base_bitset<1> &B) {
    A._M_w -= B._M_w;
}
template <size_t _Nb>
bitset<_Nb> &operator-=(bitset<_Nb> &A, const bitset<_Nb> &B) {
    _M_do_sub(A, B);
    return A;
}
template <size_t _Nb>
inline bitset<_Nb> operator-(const bitset<_Nb> &A, const bitset<_Nb> &B)
{
    bitset<_Nb> C(A);
    return C -= B;
}
template <size_t _Nw>
void _M_do_add(_Base_bitset<_Nw> &A, const _Base_bitset<_Nw> &B) {
    for (int i=0, c=0;i<_Nw;i++)
        c=_addcarry_u64(c, A._M_w[i], B._M_w[i], (ull*)&A._M_w[i]);
}
template <>
void _M_do_add(_Base_bitset<1> &A, const _Base_bitset<1> &B) {
    A._M_w += B._M_w;
}
```

```
template <size_t _Nb>
bitset<_Nb> &operator+=(bitset<_Nb> &A, const bitset<_Nb> &B) {
    _M_do_add(A, B);
    return A;
}
template <size_t _Nb>
inline bitset<_Nb> operator+(const bitset<_Nb> &A, const bitset<_Nb> &B)
{
    bitset<_Nb> C(A);
    return C += B;
}
```

Berlekamp-Massey

BOJ 11726 - 2×n 타일링

BOJ 9095 - 1, 2, 3 더하기

BOJ 9461 - 파도반 수열

BOJ 1492 - 합

```
// Usage: vl init { 2L개 이상의 초기 항 }; ll result = guess_nth_term(init, n);
// Note: 상수 계수 L차 선형 점화식의 n번째 항을 찾을 때 사용. 0-based
// O(L^2*logn)
vl berlekampMassey(vl s) {
    ll n=s.size(), L=0, m=1, d=0, coef=0;
    vl C(n), B(n), T(n);
    C[0]=B[0]=1;
    ll b=1;
    for (ll i=0;i<n;i++) {
        d=0;
        for (ll j=0;j<=L;j++)
            d=(d+C[j]*s[i-j])%MOD;
        if (d==0) {
            ++m;
            continue;
        }
        coef=d*modpow(b, MOD-2, MOD)%MOD;
        T=C;
```

```
        for (ll j=0;j+m<n;j++) {
            C[j+m]=(C[j+m]-coef*B[j])%MOD;
            if (C[j+m]<0) C[j+m]+=MOD;
        }
        if (2*L<=i) {
            L=i+1-L;
            B=T;
            b=d;
            m=1;
        }
        else {
            ++m;
        }
    }
    C.resize(L+1);
    vl tr(L);
    for (ll i=1;i<=L;++i)
        tr[i-1]=(MOD-C[i])%MOD;
    return tr;
}

ll get_nth(vl &S, vl &tr, ll k) {
    ll n=tr.size();
    auto combine=[&](vl &a, vl &b) {
        vl res(2*n+1);
        for (ll i=0;i<=n;i++)
            for (ll j=0;j<=n;j++)
                res[i+j]=(res[i+j]+a[i]*b[j])%MOD;
        for (ll i=2*n;i>n;i--)
            for (ll j=1;j<=n;j++)
                res[i-j]=(res[i-j]+res[i]*tr[j-1])%MOD;
        res.resize(n+1);
        return res;
    };

    vl pol(n+1), e(n+1);
    e[1]=1;
    pol[0]=1;
```

```
    for (++k;k>0;k>>=1) {
        if (k&1) pol=combine(pol, e);
        e=combine(e, e);
    }
    ll res=0;
    for (ll i=0;i<n;i++)
        res=(res+pol[i+1]*S[i])%MOD;
    return res;
}

ll guess_nth_term(vl x, ll n) {
    if (n<x.size()) return x[n];
    vl tr=berlekampMassey(x);
    if (tr.empty()) return x[0];
    return get_nth(x, tr, n);
}
```