

# ENOSTAVNI UREJEVLANI alg.

- ° SELECTION SORT → po vrsti izbiramo i-ti najmanjši element.
- ° INSERTION SORT → po velikosti tabele  $i=\{1 \dots n\}$  vstavljamo elemente.
- ° BUBBLE SORT → vstavimo redne pare dokler mi sortiramo, časovne raztornosti  $O(n^2)$ .

# NAPREDNI UREJEVLANI alg.

- ° MERGESORT → razdeli vstavam na polovico do  $n=1$ , nato po dva "merga".
- ° QUICKSORT → razdeli vstavam na majhne in velike glede na pivot do  $n=1 \dots$  (MERGESORT)

- ° HEAPSORT → selectionsort s hranjenjem elementov, da čas dostopa razmazujemo na  $O(n \log n)$ .
- ° COUNTING SORT → za majhne sername. i.z.  $O(n)$ , p.z.  $O(\max \text{ el.})$
- ° RADIX SORT → predvsem za stringe, tu sortira glede na črko. ostali včasnu  $O(n \log n)$

binary search → najhitrejše isk.  
HEAP-DVOJIŠKA KOPICA  
ABSTR. POD. TIPI  
AMORTIZIRANA ZAHT.



- SET (mnogočka)
- SLOVAR → {key, pair}
- ZGOŠČENA TABELA
  - ↳ regisovalna tabela

# DREVEZA

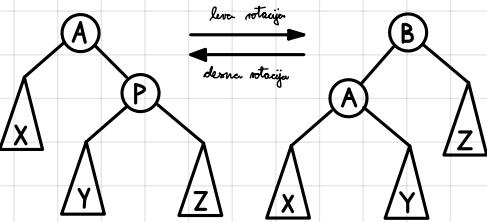
OBHODI: preni, vmesni, obratni, nisojški

VRSTE: drožidca (k-tička), polno, izkušen urejena / neurejena, črkova / ruakovna (tric).

BLOKI: število blokov  $\lceil n/B \rceil$

predobdelava  $O(n), O(n)$   
pozvedba  $O(n/B+B), O(1) \rightarrow B = \lceil \frac{n}{B} \rceil$

AVL drevesa:



```
AVLNode* AVLTree::rotateLeft(AVLNode* node) {
    AVLNode *R=node->right;
    node->right=R->left; R->left=node;
    update(node); update(R);
    return R;
```

RDEČE-ČRNA: pravna črna, obicejno mima stote, novo veličine je rdeče.

2-3 drevo: dva ali trije otroke

B-drevo: posloščina 2-3 drevera - red na pomenu največ m. otrok. (uporaba pod. last.)

NAKLJUČNO - "trap", LOMLJENO - "spay"  
POZREŠNI alg.

"v vsakem koraku se dodamo na izbiro, ki ogleda majhnejši obetavno"

ČRPALKE → če je mogoč dočeti naslednjo, posledimo  $O(n \log n)$

IZBIRA AKT. → izberemo najugodnejši konec.

REZERV. UC. →

```
VVP predavalnice(VII predavanja) {
    sort(predavanja.begin(), predavanja.end());
    VVP urnik;
    priority_queue<PII, VII, greater<PII>> pq; // min-heap
    pq.push({predavanja.back().second, -1}); // dummy
    for (auto p : predavanja) {
        auto [s,e] = p;
        auto [konec, x] = pq.top();
        if (konec==s) {
            pq.pop(); pq.push({e,x});
            urnik[x].push_back(p);
        } else {
            pq.push({e, urnik.size()});
            urnik.push_back({p});
        }
    }
    return urnik;
```

DATOTEKE na trdu: stev. velikost / pog. dobit. (dokazimo s primeri 2 dat. in  $s=1$  in  $f=1$ )

MIN. ZAMUDE: majgogradnejši red

DOKAZOVANJE PRAVILNOSTI

↳ PREDNOST (stay ahead) po vrednem koraku nester tako dobra kot opt.

↳ ZAMENJAVA (exchange alg.)

1. predpostavimo da obstaja boljša nester  $> \text{pri}$
2. arg. da bi lahko izbrali pri. nester im nester mebi bila nič slatka.

3. nameri svo protivnjake ↑ 4. □ dokazali

↳ STRUKTURA (st. arg.)

dokazemo neko struktorno lastnost optimalne nestr., ki predstavlja mijo in dokazemo, da jo posrešna nestr. dosre.

GRAFI (num., enot., cib./acib., goti./ned.)

(drevna, polni grafi, reg. grafi, dvojni grafi)

(spred, obrok, stora, pot, cikel)

(sernam gov., sre. sorodov, matrika sor.)  $C=|E|/|V|$  it. por.

(sernam gov., sre. sorodov, matrika sor.)  $C=|E|/|V|$  it. por.

PREISKOVANJE

° BFS: gre nisojški - sorodi, sorodi sorodov...

° DFS: gre povz do konca pri men sorodu.

TOPOLOŠKO UR. KRITIČNA POT

$O(n+e)$ . enak alg. obrnjen

UTEŽENI GRAFI

DIJKSTRA post.  $O(n)$ , čas.  $O(n^2)$  nar.,  $O(e \log n)$  ↗ heap

↳ iskanje najboljših poti. ↗ Bucket  $O(e+n)$

negativne vrste imajo načeloma smisel le v uravnenjih grafih.

# VPETA DREVEZA

DISJUNKTNE MN.

join  $O(1)$ , find  $O(\log n)$

MIN. VPETO DREVO "najmanjša prevera pov. vedno del min. spetege drevesa."

PRIM (poz. alg.) → izvajanje iz vrh. vrstiča

KRUSKAL (-) → dodajanje vrstiči → povravila ne smo ustvarili cicla!

i.z.  $O(mn)$ ,  $O(m \log n)$  ↗ pod. struktura disjunktne mnogočke

DELI in VLADAJ problem razdelimmo

ma več manjših podproblemov in rezultate manjših red. v nas. večjega

KROVNI IZREK → običajno različimo probleme n na  $n^{1/b}$ .  $T(n) = aT(n/b) + f(n)$

a - ist. podpr., b - vel. podpr. ?

° ENAKOMERNO RAZBITJE SEZ.

min. max. resota  $O(n \log n)$

```
partition(vector<int> a, int k) {
    int total=0, largest=0;
    for (int x : a) {
        total+=x;
        largest = max(largest, x);
    }
    int lo=largest-1, hi=total; // 1
    while (lo+1<hi) {
        int lim=(lo+hi)/2;
        int sum=0, chunks=1;
        for (int x : a) {
            if (sum+x<=lim) sum+=x;
            else { chunks++; sum=x; }
        }
        if (chunks<=k) hi=lim;
        else lo=lim;
    }
}
```

° K-TI ELEMENT  $O(n)$  ↗ C++ with default(a.begin(), ...)

DINAMIČNO prog.

lastnosti: - modrostnost podproblemov - optimalna podstrukturica večlost!

fibr., izbi. sledki, rezanje police, pot v mnaji...

NAJDALJŠE SKUPNO ZAP.

```
string LCS(string s, string t) {
    int ns=s.size(), mt=t.size();
    int lcs[n+1][m+1]; // dodatna vrstica in stolpec
    memset(lcs, 0, sizeof(lcs));
    for (int i=n-1;i>=0;i--) {
        for (int j=m-1;j>=0;j--) {
            lcs[i][j]=max(lcs[i+1][j], lcs[i][j+1]);
            if (s[i]==t[j]) lcs[i][j]=max(lcs[i][j], lcs[i][j+1]+1);
        }
    }
    // izpis izracunane tabele
    for (int i=0;i<n;i++) {
        for (int j=0;j<m;j++) {
            cout << lcs[i][j] << '\t';
        }
        cout << endl;
    }
    // rekonstrukcija
    string l="";
    int i=0, j=0;
    while (i<n && j<m) {
        if (lcs[i][j]==lcs[i+1][j]) i++;
        else if (lcs[i][j]==lcs[i][j+1]) j++;
        else { l+=s[i]; i++; j++; }
    }
    return l;
}
```

NASLEDNJA

STRAN



nahrbnik je je te snovi!

# RAČUNSKA GEOMETRIJA

**PRESEČIŠČA:** teka  $S$  in pr.  $P$ :  $P \cdot S = V_p = 0$   
 premice  $P$  in  $R$ :  $P_0 + aV_p = R_0 + bV_R$ , projekcija:  $a = \frac{a \cdot b}{\|b\|^2} b$

**POVRŠINA VEČKOTNIKA:**

osnova:  $p = \frac{1}{2} |AB| \times AC$

pri množenju koordinate razdelimo po manjšajojih  $x$  in na vsakem intervalu racinamo glavnino  $\rightarrow \sum$

**VSEBOVANOST TOČKE:**

siljamo stevi točko, če steva nado  
stevilo premic je točka vsebovana.

**KONVEKSNA OVOJNICA:**

ranjanje darila  
pri obliki je hyperbolična  
relativna produkt ...  
če je  $AC \cdot AB > 0$  enostavno  
je kot hr, drugače je dom.

↪ za sortiranje list uporabimo `list::sort()`  
identifikacija stranic

"je vse točke nahajajo na isti  
 $O(n^3, n^2, n \log n)$  strani dolje?"

```
VI toposort(vector<VI> &sosedji, int n) {
    vector<int> indeg(n);
    for (int x=0; x<n; x++) {
        for (int y : soledi[x]) indeg[y]++;
    }
    queue<int> q;
    for (int x=0; x<n; x++) {
        if (indeg[x]==0) q.push(x);
    }
    vector<int> seq;
    while (!q.empty()) {
        int x=q.front(); q.pop();
        seq.push_back(x);
        for (int y : soledi[x]) {
            indeg[y]--;
            if (indeg[y]==0) q.push(y);
        }
    }
    return seq;
}
```

## NAHRBTNIK

```
const int n = 4;
const int nosilnost = 40;
vector<int> teza = {30, 10, 40, 20};
vector<int> vrednost = {10, 20, 30, 40};

int f[n+1][nosilnost+1];
memset(f, 0, sizeof(f));
for (int i=n-1; i>=0; i--) {
    for (int x=0; x<=nosilnost; x++) {
        f[i][x] = f[i+1][x]; // ne uporabimo i-tega predmeta
        if (teza[i]<=x) { // poskusimo uporabiti i-ti predmet
            f[i][x] = max(f[i][x], vrednost[i]+f[i+1][x-teza[i]]);
        }
    }
}
cout << f[0][nosilnost] << endl;
```

↑  
topološko urejanje  
od prej ...

# C Sheet CHEAT

**STDIN/OUT** cout << " "; cin >> a,

**STRING** string str; str.size(); // velikost

**PAIR** pair<int,int> p

**AUTO** (določitev tipa) auto o; obnaša se kot tip.

**ITERATOR** for (vector<int>::iterator it=v.begin(); it != v.end(); ++it)

**RAZPAKIRANJE** for (auto [x,y]: pairs) cout << x << y

**VRSTA** queue<T>

front(); back(); push(n); pop(); empty(); ...

**SKLAD** stack<T>

top(); push(n); pop(); empty(); ...

**MNOZICA** map<string,int>

begin(); end(); insert(...); erase(...); find(...); /\* -key \*/ count(...); ...

**POV. SEZNAM** list<T>

front(); back(); empty(); push\_front(); push\_back(); insert(...); ...

**REFERENCE**

```
auto podvoji(vector<int> v) {
    int n=v.size();
    for (int i=0; i<n; i++) {
        v.push_back(v[i]);
    }
    return v;
}
```

← ta funkcija me spremeni vrednosti prototipa seznama!  
ta pa jo →

```
void podvoji_ref(vector<int> &v) {
    int n=v.size();
    for (int i=0; i<n; i++) {
        v.push_back(v[i]);
    }
}
```

f(int& poz) poz=0; // u get the idea ...

**MIN, MAX** min(5,2); max({5,2,0});

\*min\_element(v.begin(), v.end());

**SORT**

sort(v.begin(), v.end());

**ANONIMNE** funkcije

**SEZNAM** vector<T>

resize(n); capacity(); push\_back(n); pop\_back(); clear(); swap(v); ...

```
sort(sez.begin(), sez.end(), [](int a, int b) {
    return to_string(a) < to_string(b);
});
```

