

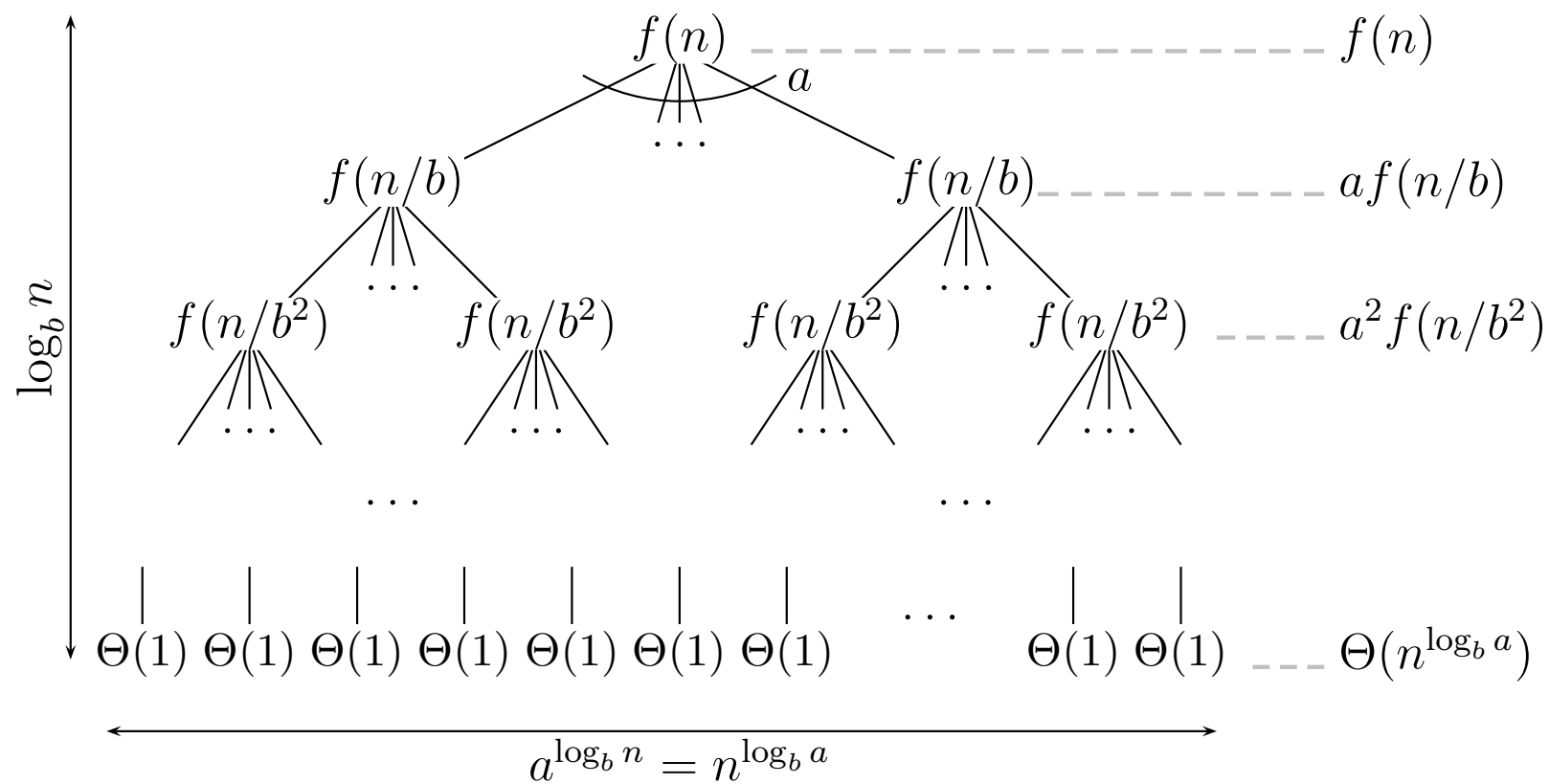
Master theorem:

Nech $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$. Nech $k = \log_b a$. Potom:

1. Ak $f(n) \in O(n^{k-\varepsilon})$ pre niektoré $\varepsilon > 0$, potom $T(n) \in \Theta(n^k)$.
2. Ak $f(n) \in \Theta(n^k)$, potom $T(n) \in \Theta(f(n) \log n)$.
3. Ak $f(n) \in \Omega(n^{k+\varepsilon})$ pre niektoré $\varepsilon > 0$ a platí podmienka regularity, potom $T(n) \in \Theta(f(n))$.

Podmienka regularity: Existuje $c < 1$ také, že pre všetky dostatočne veľké n platí $af(n/b) \leq cf(n)$.

Poznámka: Veta platí aj v prípade rozumných usporiadaní dolných a horných celých častí - vid' napr. CLRS2 4.4.2.



```

function closest_pair(l,r)
    // Find the closest pair in P[l..r]
    // assume P[l..r] is sorted by x-coordinate
    if size(P)<2 then return infinity;
    // Divide: midx will be a dividing line
    mid:=(l+r)/2; midx:=P[mid].x;
    dl:=closest_pair(l,mid); dr:=closest_pair(mid+1,r);
    // as a side effect, P[l..mid] and P[mid+1..r]
    // are now sorted by y-coordinate
    delta:=min(dl,dr);
    QL:=select_candidates(l,mid,delta,midx);
    QR:=select_candidates(mid+1,r,delta,midx);
    dm:=delta_m(QL,QR,delta);
    // use merge make P[l..r] sorted by y-coordinate
    merge(l,mid,r);
    return min(dm,dl,dr);

```

```
function select_candidates(l,r,delta,midx)
    // From P[l..r] select all points which are
    // in the distance at most delta from midx line
    create empty array Q;
    for i:=l to r do
        if (abs(P[i].x-midx)<=delta)
            add P[i] to Q;
    return Q;
```

```

function delta_m(QL,QR,delta)
    // Are there two points p in QL, q in QR such that
    // d(p,q)<=delta? Return closest such pair.
    // Assume QL and QR are sorted by y coordinate
    j:=1; dm:=delta;
    for i:=1 to size(QL) do
        p:=QL[i];
        // find the bottom-most candidate from QR
        while (j<=n and QR[j].y<p.y-delta) do
            j:=j+1;
        // check all candidates from QR starting with j
        k:=j;
        while (k<=n and QR[k].y<=p.y+delta) do
            dm:=min(dm,d(p,QR[k]));
            k:=k+1;
    return dm;

```

```
//----- main -----  
// P contains all the points  
sort P by x-coordinate;  
return closest_pair(1,n);
```

Časová zložitosť: Nech $T(n)$ je čas potrebný na vyriešenie problému pre n bodov.

- Rozdeľuj: $\Theta(1)$
- Panuj: $2T(n/2)$
- Skombinuj: $\Theta(n)$

Teda $T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n)$ (master theorem).

Opakovanie: Greedy algoritmy

- v každom kroku vezmeme lokálne optimálny krok
- ľahké na implementáciu
- obvykle veľmi efektívne (časová zložitosť)
- metóda sa často nedá použiť
- hlavný problém: **dokázať správnosť**

Príklady použitia:

- Výber aktivít $\Theta(n \log n)$
- Najlepšie Huffmanovo kódovanie $\Theta(n \log n)$
- Rozmieňanie peňazí (niektoré systémy) $\Theta(m)$

Opakovanie: Dynamické programovanie

- rozkladáme problém na podproblémy
- počítame optimálne riešenia pomocou rekurencií vo veľkej matici podproblémov
- ľahké na implementáciu
- hlavný problém: vymyslieť správny podproblém

Príklady použitia:

- Rozmieňanie peňazí (všeobecné) $\Theta(mS)$
- Najdlhšia spoločná podpostupnosť $\Theta(mn)$
- Najkratšia triangulácia $\Theta(n^3)$

Opakovanie: Rozdeľuj a panuj

- rozdeľ problém na menšie podproblémy, vyrieš rekurzívne a skombinuj čiastkové riešenia
- efektívne
- niekedy ťažké na implementáciu, veľký overhead na rekurziu
- hlavný problém: **analýza časovej zložitosti**

Príklady použitia:

- Triedenie (merge sort, quick sort) $\Theta(n \log n)$
- Násobenie veľkých čísel $\Theta(n^{1.58\dots})$
- Násobenie matic (Strassenov algoritmus) $\Theta(n^{2.81\dots})$
- Najbližší pár bodov $\Theta(n \log n)$
- Výber k -teho prvku $\Theta(n)$