Programm:

1. Kombinatorische Probleme
2. Divide & Conquer
   a) Mergesort
   b) Dichtestes Punktepaar

Kombinatorische Probleme

Ganz allgemein: Gegeben ein Objekt M, (in Abhängigkeit von M)
die Beschreibung einer endlichen! Lösungsmenge L,
ein Prädikat $P: L \to \{true, false\}$ und
eine Funktion $f: L \to \mathbb{R}$

Gesucht: Eine Lösung $x \in L$ mit $P(x)$ (=true) und $f(x)$ minimal.

Lösung ist „zulässig"

Algorithmus (Brute-Force):

```
val := ∞ / opt := NULL
for each x ∈ L {
    if P(x) ∧ f(x) < val {
        opt := x
        val := f(x)
    }
}
return opt
```

Laufzeit? $\Theta(|L|)$

Beispiel 1: Sortieren

- M ist ein Array der Länge n
- L ist die Menge aller Permutationen von M
- $P(x) = true \iff x_0 \leq x_1 \leq \dots \leq x_{n-1}$
- $f(x) = 0$ für alle $x \in L$

Kardinalität von L? $|L| = n!$

Beispiel 2: Kürzester Weg in einem Graphen zwischen p und q

- $M = (V, E)$
- $L = 2^E$ (die Potenzmenge von E)
- $P(x) = true \iff x$ ist ein Weg von p nach q
- $f(x) = |x|$

$\hookrightarrow |L| = 2^{|E|}$

Beispiel 3: TSP

Gegeben: ein vollständiger Graph $G=(V,E)$ und eine Funktion $c: E \to \mathbb{R}^+$.

Gesucht: eine kürzeste Rundreise

$\hookrightarrow$
- $M = (V,E)$
- L ist die Menge aller Permutationen von V
- $P(x) = \text{true}$ für alle $x \in L$
- $f(x) = \sum_{i=0}^{|V|-1} c(\{x_i, x_{(i+1) \bmod n}\})$

$\hookrightarrow$ $|L| = |V|!$

Wie schnell sind gute Algorithmen dafür?

Bsp. 1.: Mergesort $\Theta(n \log n)$

Bsp. 2.: BFS $\Theta(|V| + |E|)$

Bsp. 3.: ? (bekannt: $O(2^n n^2)$ mittels DP)

Entscheidend ist nicht $|L|$, sondern die Struktur von $L' = \{x \in L \mid P(x)\}$!

Intuition:
- Beim Sortieren kann man z.B. Transitivität nutzen – dadurch funktioniert Merge, noch offensichtlicher ist es bei Quicksort (bald in der VL!).
- Für kürzeste Wege gilt u.a., dass jeder Teilweg eines kürzesten Weges selbst ein kürzester Weg ist.

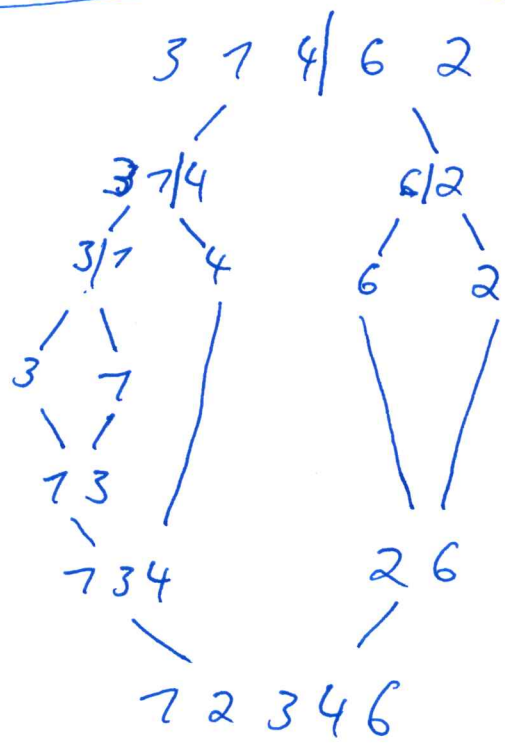
Es ist nicht klar, ob sich die Struktur von TSP-Lösungen ähnlich gut fassen lässt (Vermutung: Nein).

Divide & Conquer

Idee: - Kleine Probleminstanzen können direkt gelöst werden.
- Große Probleminstanzen in kleine aufteilen, (divide)
rekursiv lösen, } (conquer) (oder conquer + combine)
Lösungen zusammenfügen.

Mergesort (*)

```
Mergesort(A,p,r) {
    if p<r {
        q := ⌊(p+r)/2⌋
          Mergesort(A,p,q)
          Mergesort(A,q+1,r)
          Merge(A,p,q,r)
    }
}
```



Das letzte Merge genauer:

A: 1 3 4 2 6

Merge(A, 1, 3, 5)

$n_1 := q-p+1 = 3$, $n_2 := r-q = 2$

L: 1 3 4 ∞ , R: 2 6 ∞

A: 1 2 3 4 6

(*) Grundidee: - Arrays der Länge ≤ 1 sind bereits sortiert
- Zwei sortierte ($\lceil r/2 \rceil$) Arrays lassen sich leicht zu einem sortierten Gesamtarray zusammenfügen

↳ Also: D&C

Laufzeit von Mergesort:

$$T(n) = \begin{cases} O(1) & \text{falls } n = 1 \\ 2T(\frac{n}{2}) + O(n) & \text{falls } n \geq 2 \end{cases}$$

Lösung aus der VL: $T(n) \in O(n \log n)$

(optimal für vergleichsbasiertes Sortieren)

Dichtestes Punktepaar

Gegeben: Menge $V = \{p_1, p_2, \dots, p_n\}$ von Punkten in der Ebene, $p_i = (x_i, y_i)$

Gesucht: $\min_{p_i, p_j \in P,\, p_i \neq p_j} d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ minimal.

Nach der Definition vom Anfang:

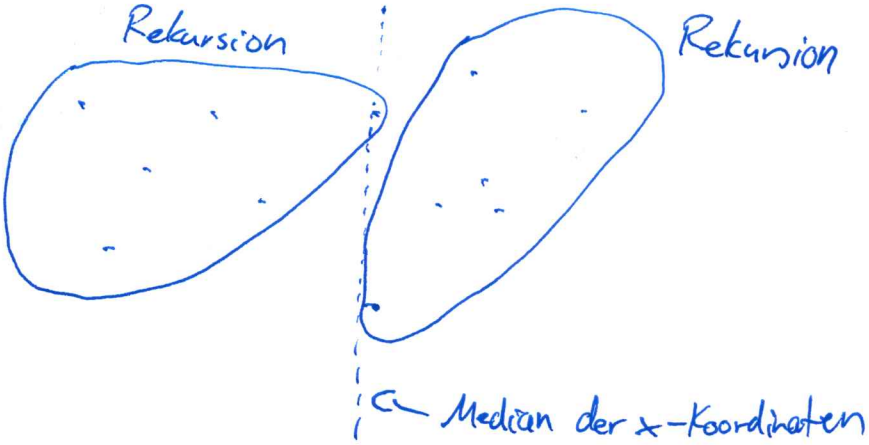
- $M = V$
- $L = \{(p_i, p_j) \mid p_i \in M, p_j \in M, p_i \neq p_j\}$
- $P(x) = \text{true}$ für alle $x \in L$
- $f(x) = x$ für

$\hookrightarrow |L| = \binom{n}{2} = \frac{n(n-1)}{2} \in O(n^2)$

Somit hat Brute-Force Laufzeit $O(n^2)$ für das Problem.

Geht es schneller? Ja!

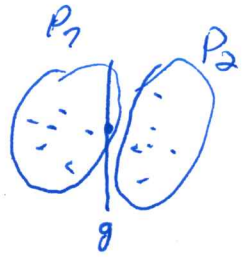
Idee:



~~ABER: Was passiert, wenn das dichteste Punktepaar durch die Gerade getrennt wird?~~

Alg. DichtestesPaar(P):

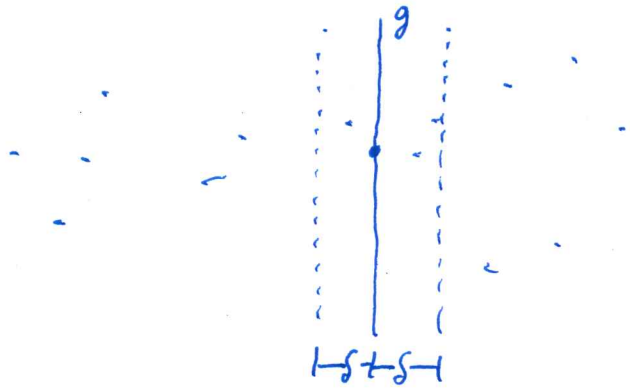
1. if $|P| = 1$: return ∞
2. Finde Median der x-Koordinaten x_m
~~Sortiere P nach x-Koordinaten~~
3. Teile P mit einer vertikalen Gerade g durch ~~dem Median der x-Koordinaten~~ x_m in Mengen P_1 und P_2



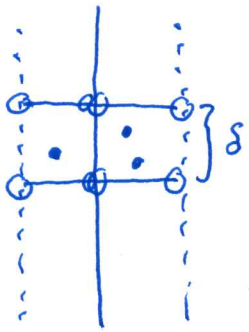
4. $\delta_1 :=$ DichtestesPaar(P_1)
5. $\delta_2 :=$ DichtestesPaar(P_2)
6. $\delta_{12} :=$ kürzeste Entfernung zwischen Punkten in P_1 und Punkten in P_2
7. return $\min\{\delta_1, \delta_2, \delta_{12}\}$ (*) Zu Schritt 2...

Spannend bleibt Schritt 6: Das sind $\frac{n}{2} \cdot \frac{n}{2} \in \Omega(n^2)$ viele Paare! Struktur hilft!

Sei $\delta := \min\{\delta_1, \delta_2\}$



Es müssen nur Punkte in einem Streifen der Breite 2δ um g getestet werden. Außerdem müssen die Punkte auch in y-Richtung nah beieinander sein.



~~Höchstens~~ Die

Höchstens die 8 nächsten Punkte in y-Richtung kommen dafür infrage.

Mit einer Vorsortierung der Punkte nach y-Koordinate kann Schritt 6 in Zeit $O(n)$ ausgeführt werden.

Für die Laufzeit ergibt sich somit (für $n = |P|$)

$$T(n) = \underbrace{O(n \log n)}_{\text{Sortierung } x} + \underbrace{O(n \log n)}_{\text{Sortierung } y} + \underbrace{T'(n)}_{\text{Rekursion}}$$

$$\text{mit } T'(n) = \left\{ \underbrace{2T\left(\frac{n}{2}\right)}_{\text{Rekursion}} + \underbrace{O(n)}_{\text{Combine}} \begin{matrix} : n = 1 \\ : \text{sonst} \end{matrix} \right\} \in O(n \log n) \; *$$

$O(1)$

Also: $T(n) \in O(n \log n)$.

(*) Zu Schritt 2:

Der Median kann leicht mit Sortieren in $O(n \log n)$ bestimmt werden, was für unsere Zwecke reicht. Es geht aber auch schneller – mehr dazu bald in der VL!