Algorithm Theory - Exercise Class

Exercise Lesson 1

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Philipp Schneider

Algorithms and Complexity - Professor Dr. Fabian Kuhn

Exercise 3

Let α, β be *constants* let $T(n)$ be *monotonously increasing* with

$$T(1) \leq \alpha, \quad T(n) \leq 4 \cdot T(n/4) + \beta \cdot n.$$

Give an *asymptotic* upper bound ($\mathcal{O}$-Notation) for $T(n)$, assuming $n = 4^k$ for some $k \in \mathbb{N}$. Prove that the same bound applies to $T(n)$ for *all* $n \in \mathbb{N}$ (i.e., if n is not necessarily a power of 4).

$T(n) \in \Theta(n \log n)$ for $n = 4^k$, $k \in \mathbb{N}$

1. Solution:

Let $n \in \mathbb{N}$, $4^{k-1} < n \leq 4^k$

$$\begin{aligned} T(n) &\leq T(4^k) \\ &\leq 4 \cdot T(n/4) + \beta \cdot n \\ &\leq 4 \cdot T(4^{k-1}) + \beta \cdot n \\ &= 4 \cdot (\alpha 4^{k-1} + \beta 4^{k-1}(k-1) + \beta 4^{k-1} \\ &\leq 4 \cdot (\alpha n + \beta n \log_4 n + \beta n) \in \Theta(n \log n) \end{aligned}$$


UNI
FREIBURG

Exercise 5

UNI FREIBURG

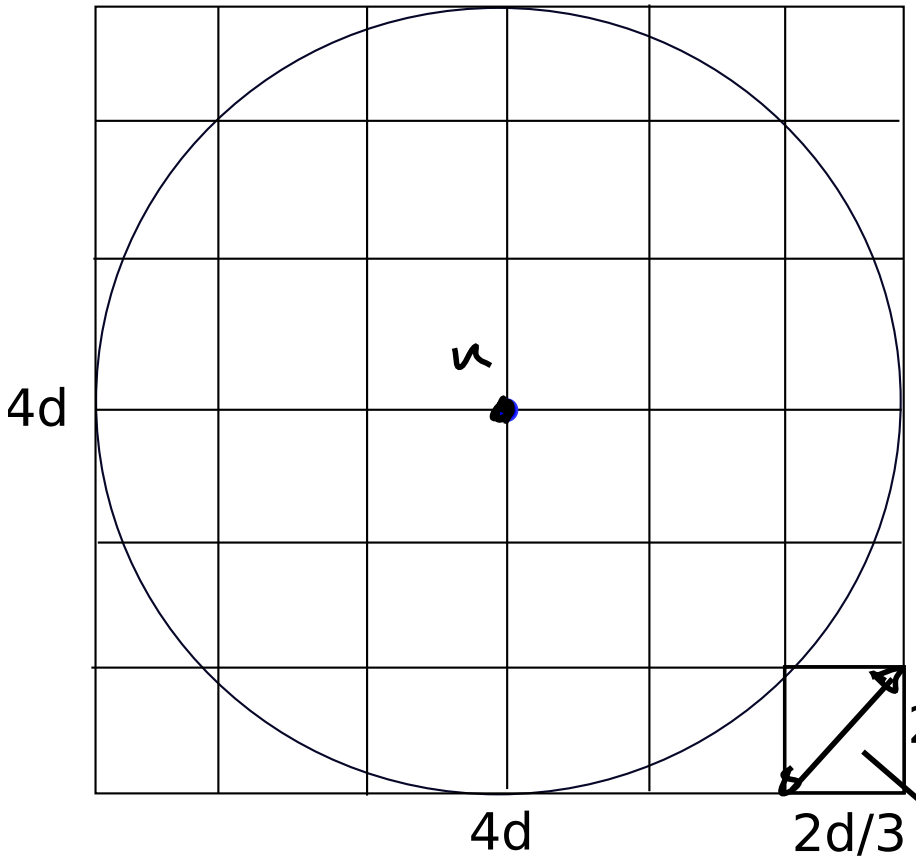
In the lecture, we discussed an $\mathcal{O}(n \log n)$-time divide-and-conquer algorithm to determine the closest pair of points among a set of n points on the real plane $\mathbb{R}^2$. Assume that we are not only interested in the closest pair of points, but in **all pairs of points that are at distance at most twice the distance between the closest two points**.

(a) How many such pairs can there be? Give a nontrivial answer. It suffices to give your answer using the $\mathcal{O}$-notation. Proof would be nice too.

(b) Devise an algorithm that outputs a list with all pairs of points at distance at most twice the distance between the closest two points and write it down. Describe what you have to change compared to the closest pair algorithm of the lecture and analyze the running time of your algorithm.

Exercise 5 (a)

$$d = \min_{u,v \in S} d(u,v)$$



max. 36 Punkte innerhalb 2d Radius um u.

Insgesamt weniger als $36 \cdot n \in O(n)$ Paare mit Abstand $\leq 2d$

$$\frac{\sqrt{2}\,2d}{3} < d$$

Exercise 5 (b)

Algorithm 1 DoubleMinDist(S) ▷ returns (d, L, Y)

▷ *Note:* $d = \min_{u,v \in S} d(u,v)$, $L = \{(u,v) \mid u,v \in S, d(u,v) < 2d\}$, $Y =$ *"S sorted by y-coordinate"*

if $|S| < 4$ **then** ▷ *Base case,* $\mathcal{O}(1)$
  Use the naive Algorithm to calculate (d, L, Y)
  return (d, L, Y)

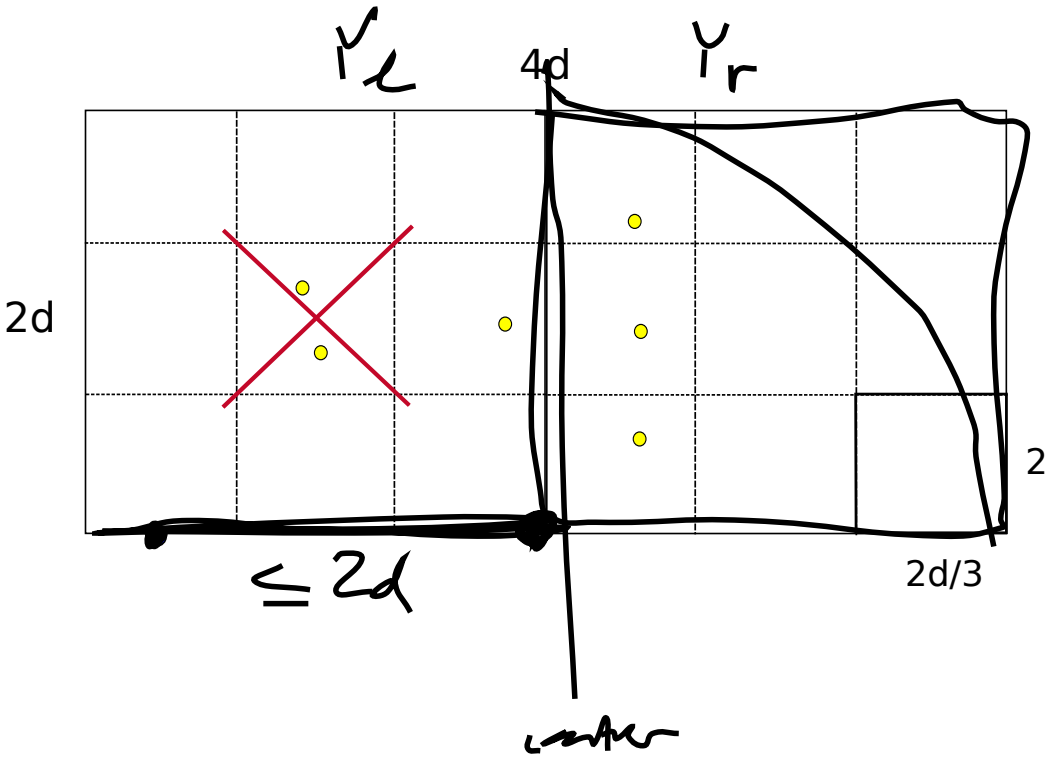
Divide S equally into S_ℓ and S_r along a vertical axis ▷ *"Divide",* $\mathcal{O}(n)$

$(d_\ell, L_\ell, Y_\ell) \leftarrow$ DoubleMinDist(S_ℓ) ▷ *"Conquer",* $2 \cdot T(n/2)$
$(d_r, L_r, Y_r) \leftarrow$ DoubleMinDist(S_r)

▷ *"Merge",* $\mathcal{O}(n)$

$d_{\ell r} \leftarrow \min_{u \in Y_\ell, v \in Y_r} d(u,v)$ ▷ *Possible in* $\mathcal{O}(n)$ *(cf. explanation below)*
$L_{\ell r} \leftarrow \{(u,v) \mid u \in Y_\ell, v \in Y_r, d(u,v) < 2d\}$ ▷ *Possible in* $\mathcal{O}(n)$ *(cf. explanation below)*
$d \leftarrow \min(d_\ell, d_r, d_{\ell r})$ ▷ *Calculate new minimum distance,* $\mathcal{O}(1)$
$L \leftarrow L_\ell \cup L_r \cup L_{\ell r}$ ▷ *Concatenate lists,* $\mathcal{O}(n)$
$L \leftarrow L \setminus \{(u,v) \mid d(u,v) > 2d\}$ ▷ *Remove pairs violating new minimum distance 2d,* $\mathcal{O}(n)$
$Y \leftarrow$ Merge(Y_ℓ, Y_r) ▷ *Merge sorted lists into one,* $\mathcal{O}(n)$ *(cf. Mergesort)*
return (d, L, Y)

Exercise 5 (b)

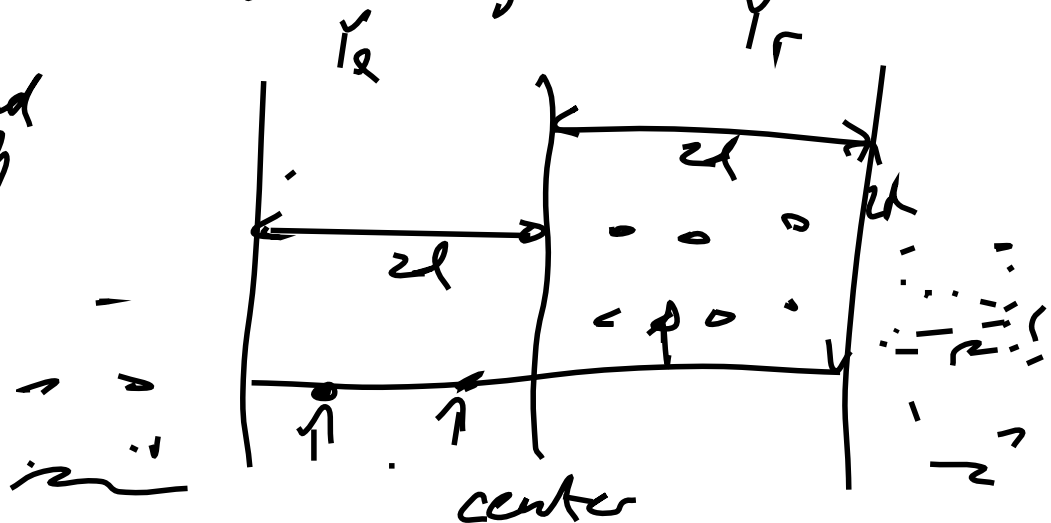


jeder Punkt u mit Abstand $\leq 2d$ von der "Trennlinie" in Y_L kann höchstens 9 Punkte in Y_r haben die Abstand $\leq 2d$ von u und eine y-Wert größer als u haben.

Und umgekehrt.

$Y'_L = Y_L \setminus \{$ Punkte mit Abstand $> 2d$ zu center $\}$

$Y'_R = Y_R \setminus$ —— u ——




UNI
FREIBURG

Fractional Knapsack problem

UNI FREIBURG

In the fractional Knapsack problem a thief robbing a store finds n items. The i-th item is worth v_i euro and weighs w_i kilograms. The thief wants to take a load which is as valuable as possible, but he can carry at most W kilograms in his knapsack. However, the thief can take fractions of items. $v_i, w_i > 0$

(a) Formulate a simple greedy algorithm running in $\mathcal{O}(n \log n)$

(b) Show that this greedy algorithm computes an optimal solution.

(c) Assume that you have an algorithm MEDIAN that computes the median of a list in $\mathcal{O}(n)$ time. Use this to improve the runtime to $\mathcal{O}(n)$.

GreedyThief $((v_1, w_1), \ldots, (v_n, w_n), W)$

- sort items by v_i/w_i (w log now sorted) $O(n \log n)$
- while $(W > 0)$
  - Steal as much of j as possible (x_j)
  - $W \leftarrow W - x_j w_j$
  - $j \leftarrow j + 1$

} $O(n)$

$S^* = (y_1, \dots, y_n)$ ↳ Opt

$S = (x_1, \dots, x_n)$ ↳ Greedy solution

wlog $v_1/w_1 \geq \dots \geq v_n/w_n$ sorted (descending)

wlog $v_i/w_i \neq v_j/w_j$ for $i \neq j$ ⟸

Let k be the smallest value with $x_k \neq y_k$

$\Rightarrow x_k \geq y_k$

Angenommen $x_k > y_k$

Dann sei $\delta = x_k - y_k$ dann entferne aus S^* von items l, $l \geq k$ die gleiche Menge δ, wodurch S^* sich verschlechtert ↯

$\Rightarrow x_k = y_k \quad \Rightarrow S = S^*$

FastThief (List of indices, W)

m ← Pick median of list $(v_1/w_1, \ldots, v_n/w_n)$ ← O(n)

Compute $G = \{ i \mid v_i/w_i > m \}$

$L = \{ i \mid \text{--''--} < \text{--''--} \}$

$E = \{ i \mid \text{--''--} = \text{--''--} \}$

} O(n)

if ($w_G \leq W$) ←

  Steal G ←

  if ($w_G + w_E \geq W$)

    Steal as much of E as possible

  else ($w_G + w_E < W$)

    steal E

    FastThief (L, $W - w_G - w_E$)

else ($w_G > W$)

  FastThief (G, W)

$T(n) \leq T(\frac{n}{2}) + O(n)$

M.T. $\Rightarrow$ $O(n)$

UNI
FREIBURG