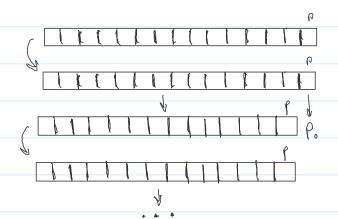
AuD Hausübung 3 Nicolas Petermann, Julian Imhof, Jonas Franz

H1) Die Laufzeit warde OCn2) betragen, da in jeder Iteration and jedem Teilanay unabhängig vorm gewählben Plust Element Ceine Elemente existieren, welche größer als das Pivol- Element sind. Somit bleibt dus Pivot-Flement in jeder Ite-lian an seiner ursprungliden Stehle, in diesem Fall reals, was bedeutet, dass sich die Array-Länge Pro Iteration sid nur um 1 reduziet. Somit haben wir einen Listenanhliden Relusionsbaum:



b) A= [16, 3, 5, 20, 7, 2, 19, 8, 20, 17, M] [3 5 10 7 2 8 M 19 20 19 6] [35 10 728] [19 70 N7 16] (E35 728) 70] (E16) 19 20 17] [369 1359

Bester Fall: A-[2,3,7,10,5,16,17,20,19,11] Anzahi Vergleiche: 36 Schlechtester Fall: B=[20, 19, 17, 16, 11, 10, 7, 5,3,2] Anzain Vergleiche: 45 (12) A) findIndex(A) if (A.length == 1)return 0 if (A.length > 1)if (A [0] > A [1]) return 0 if (A [A.length - 1] > A [A.length - 2]) return A.length - 1int 1 = 0int r = A.lengthwhile (r - l > 1) do int mid = floor((1 + r) / 2)if (A [mid] > A [mid + 1])r = midelse l = midreturn r Korrelcheit: Schleifeninvariante: 3ª jedem Zeitpunkt befindet Sich der gesuchte Index! innerhalb von I, r, sodass

Sich der gesuchte Index! innerhalb von I, r, sodass

1 = i = r.

Schleifeninvariante gilt vor der Schleife: Da A II. 12 gesamtes Array abdedet muss k innerhalb von 1, r liegen.

In jeder Iteration wird entweden I oder reschoben. Zunächst wird mid mit L(Itr)/2]
berechnet. mid tepräsentiert den index in der Mitte zwischen I und r. Per abfrage wird entschieden, ob I oder r verschoben werden Soli. Angenommen, der Welt in a an index mid ist größer als der Wet an 'dex midtn, so muss der gesamte Teilanay A [mid] Ichehar gleich A [mid] sein. Somit muss der gesuchte index Im Bereich A [1, mid] Ilegan. Sollte dieser

Fall einfeter, verschieben wir r zu Mid.

Somet muss land oben der gesuchte index

im never Teilarray liegen.

Invariante quit nach der Schleife:

Die Schleife terminiert gdw. dar Teilarray nur

Noch aus einem Wert besteht. Ober haber wir

gezeigt, dass der gesuchte index dar Teilarray

nich vertakst. Also muss er sich bet dem

```
nich vertokat. Also muss es sich bet dem
letyla Element um der index handeln.
Loufzeit:
Da sich die Lange der Arroys in jeder
Iteration halbiert, eshalter wir so eine
Laufzeit von O(log n).
 findIndex(V)
     if (V.length == 1) return 0
     if (V.length > 1)
        if (V [0] > V [1]) return 0
         if (V [V.length - 1] > V [V.length - 2]) return
 V.length - 1
     int 1 = 0
     int r = V.length
     while (r - 1 > 1) do
         int mid = floor((1 + r) / 2)
         if (V [mid] > V [mid + 1])r = mid
        else 1 = mid
     return V [r]
 findIndex(W)
     if (W.length == 1)return 0
     if (W.length > 1)
        if (W [0] > W [1]) return 0
         if (W [W.length - 1] > W [W.length - 2])return
 W.length - 1
     int 1 = 0
     int r = W.length
     while (r - 1 > 1) do
        int mid = floor((1 + r) / 2)
         if (W [mid] > W [mid + 1])r = mid
        else l = mid
     return W [r]
```

H3)

6

		tr	ans f lgf	iti uni Polg	ai v d uc	us () (x h	Eig Ka 4 Ka 4	X	2) 3)	ch 1 1	af () (y (y	te	n	(3))		a
		fo fo	o lg ans f lgf	iti uni Polg	ai v d uc	us () (x h	Eig Ka 4 Ka 4	X	2) 3)	ch 1 1	af () (y (y	te	n	(3))		
		fo fo	o lg ans f lgf	iti uni Polg	ai v d uc	us () (x h	Eig Ka 4 Ka 4	X	2) 3)	ch 1 1	af () (y (y	te	n	(3))		•
		fo fo	o lg ans f lgf	iti uni Polg	ai v d uc	us () (x h	Eig Ka 4 Ka 4	X	2) 3)	ch 1 1	af () (y (y	te	n	(3))		n
		tr	ans f lgf	un Volg	v d nt uc	(x h	x = 5	X	e) e)	1	(y (y	1 2 2	 	(3) (3))		
•		fo	f lgt	un Polg	d et uc	Cx h	(a <u>s</u>	X	,)	1	(y (y	1 5	≤ } ≤ }	3)	vo	
		+	f Igt	olg	rt uc	(x h	1 5	X	3)	1	Cy	1 5	i y	/3		vo	
•		+	f Igt	olg	rt uc	(x h	1 5	X	3)	1	Cy	1 5	i y	/3		vo	•
•		+	lg f	a	uc	4					-		-			vo.	7
,		+	+	+		+	au	5	t	ig	en	scl	naj	16	n	vo.	1
)		an	tis	/ 10				+-			-	-		-	1		1
)		an	1151	100	1		-	-				1				-	1
1	to market	1 1	1		me											T	
2 3						V	ind	(×a	4	X)/	1 (y	1	y	.)
				+		20	16 F	0	20	=	χp)/	2 (V-	1 =	1/2	2)
								1									
	Da	d	ie	Re	la	Fi	on	Γ	e	le	xi	1	tr	an	sit	iv	u
	an	tis	y m	m	e /	c	sch	is	F	1	20	(g/	d	as	5 5	ie.	ei
							.	1				+			-	-	-
								1				4	7	17			
	Va															-	-
		0	de	•	(1	٢	2)	1	(2 4	1)			
		Li		P	=(2	1)	u	n	d	C	2 =	(1.	2)	-	-
	, ,			1 1										-	-	1	
Name and Address of the Party o)	an Or Da	antis Ordn Da w	Ordnun Ordnun Da wed ode	antisymm Ordnungsi Da weder oder Für P	Da die Rela antisymmet Ordnungsret Da weder (oder (tür P=(Da die Relati antisymmetri Ordnungsrela Da weder (*) für P=(2,	Da die Relation antisymmetrisch Ordnungsrelatio Da weder (12 oder (12 für P=(2,1)	Da die Relation rantisymmetrisch is Ordnungsrelation Da weder (*251) oder (*251) für P=(2,1)	Da die Relation res antisymmetrisch ist Ordnungsrelation i Da weder (*251) oder (152) für P=(2,1) un	Da die Relation refle antisymmetrisch ist f Ordnungsrelation ist Da weder (*241) 1 oder (142)1 für P=(2,1) und	Da die Relation reflexion antisymmetrisch ist for Ordnungsrelation ist. Da weder (122) n (14) oder (122) n (14) oder (122) n (14)	Da die Relation reflexiv, antisymmetrisch ist folgt Ordnungsrelation ist Da weder (121) 1(1) oder (122) 1(2) für P=(2,1) und Q=	Da die Relation reflexiv, tra antisymmetrisch ist folgt, d Ordnungsrelation ist Da weder (122)1 (142 oder (122)1 und Q=(Da die Relation reflexiv, tran antisymmetrisch ist folgt, das Ordnungsrelation ist Da weder (121) 112) oder (121) 112) für 112	Da die Relation reflexiv, transit antisymmetrisch ist folgt, dass s Ordnungsrelation ist Da weder ($1 \leq 2$) $1 \leq 2$) oder ($1 \leq 2$) $1 \leq 2$) tür $1 \leq 2$, $1 \leq 2$) and $1 \leq 2$	Da die Relation reflexiv, transitiv antisymmetrisch ist folgt, dass sie Ordnungsrelation ist Da weder (*251) 1 (152) oder (152) 1 (251)

```
6
      List<Point> findMaxRec(List<Point> points) {
          if (points.size() = 1)return points;
          else {
              //Divide
             List<Point> p0 = new ArrayList(), p1 = new ArrayList();
              for (int i = 0; i < points.size(); i++) {
                 if (i < points.size() / 2)p0.add(points.get(i));</pre>
                 else pl.add(points.get(i));
              }
             List<Point> m0 = findMaxRec(p0), m1 = findMaxRec(p1);
              //Conquer
              List<Point> max = new ArrayList();
             m0.stream().forEach(x -> max.add(x));
             m1.stream().forEach(x -> max.add(x));
             List<Point> max0 = new ArrayList(max);
             \max.stream().forEach(x -> {
                 \max.stream().filter(y \rightarrow !y.equals(x)).forEach(y \rightarrow {
                     if (x.x \le y.x \&\& x.y \le y.y) max0.remove(x);
                 });
              });
             return max0;
       //Alternative nicht-rekursive Lösung
      List<Point> findMax(List<Point> p) {
          List<List<Point>> c = new List(p);
          List<Point> r = new List();
          r.addAll(c.get(0));
          c.get(0).fe(x -> {
              c.get(0).filter(y \rightarrow !y.equals(x)).fe(y \rightarrow {
                 if (x.x \le y.x & x.y \le y.y)r.vpop(x);
              });
          });
          return r;
c) Schleifeninvariante:
      Die Zurückgegebene Liste authölt nur maximale
Punkte.
   Vor der Schleife:
Bis die Liste nur noch ein Element hat wird
```

Bis die Liste nur noch ein Element hat wird sie tekursiv geteilt. Bei Rekursion balbruch ist die Liste Cherementig, der punkt ist also maximal.

Die beiden zurückgegeberen Listen entholter aus dem Vorigen Schrifte ebenfalls leire moximorien Pankte.

In der Schleiß:
Während jedem Rekursionsschrift werden die
zwei zurückgegeberen Listen zusammengeführt,
Die einzelnen Listen enthalbe hur Maximale
Punkte, die Vereinte Liste Lann aber num
nicht-max. Enthalten. Diese Werden vor der
Rückgabe entfernt. Die Rückgabe besteht also
nur aus maximaler Punkter aus den beiden Teilisten.

Nach der Schreife: Die letzte Ridegabe vereint die erste und zweite Listenhälfte und ontfernt wie oben dargestellt nicht-max. Die letzte Ridegabe endhält also Obenfalls keine maximaler Punkte.

Laufzeit: Liste wird in jeder Iteration halbiert

- Zusammenfüger eizewat aurah dappelte

for Schleife

NEA

