Name:
Vorname:
Matrikelnummer:

Klausur-ID:

Lösungsvorschlag

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders 13.09.2018

Klausur Algorithmen II

Aufgabe 1.	Kleinaufgaben	13 Punkte
Aufgabe 2.	Maximum Weight Matchings	9 Punkte
Aufgabe 3.	Zufallsgraphen	8 Punkte
Aufgabe 4.	Flussalgorithmen	11 Punkte
Aufgabe 5.	External Memory RMQ	12 Punkte
Aufgabe 6.	Dominierende Rechtecke	7 Punkte

Bitte beachten Sie:

- Als Hilfsmittel ist nur **ein** DIN-A4 Blatt mit Ihren **handschriftlichen** Notizen zugelassen.
- **Schreiben** Sie auf **alle** Blätter der Klausur und Zusatzblätter Ihre **Klausur-ID**.
- Merken Sie sich Ihre **Klausur-ID** auf dem Aufkleber für den Notenaushang.
- Die Klausur enthält **18 Blätter**.
- Zum Bestehen der Klausur sind 20 Punkte hinreichend.

Lösungsvorschlag

Aufgabe 1. Kleinaufgaben [13 Punkte]

a. Gegeben sei die Burrows-Wheeler-Transformation $T^{BWT} = \texttt{auu\$caihhh}$. Berechnen sie den Originaltext T. Machen Sie dabei ihren Rechenweg deutlich. [3 Punkte]

Lösung

Zunächst berechnen wir LF, indem wir die Zeichen in T^{BWT} (stabil) sortieren:

i	1	2	3	4	5	6	7	8	9	10
T^{BWT}	a	u	u	$	c	a	i	h	h	h
LF	2	9	10	1	4	3	8	5	6	7

Beginnend bei `$` können wir damit T von hinten nach vorne rekonstruieren: Das Zeichen vor `$` ist `a`, da $T^{BWT}[4] = \texttt{\$}$, $LF[4] = 1$ und $T^{BWT}[1] = \texttt{a}$. Weiter ist $LF[1] = 2$ und $T^{BWT}[2] = \texttt{u}$, weshalb das Zeichen vor `a` ein `u` ist. Und so weiter...
$T = \texttt{chihuahua\$}$

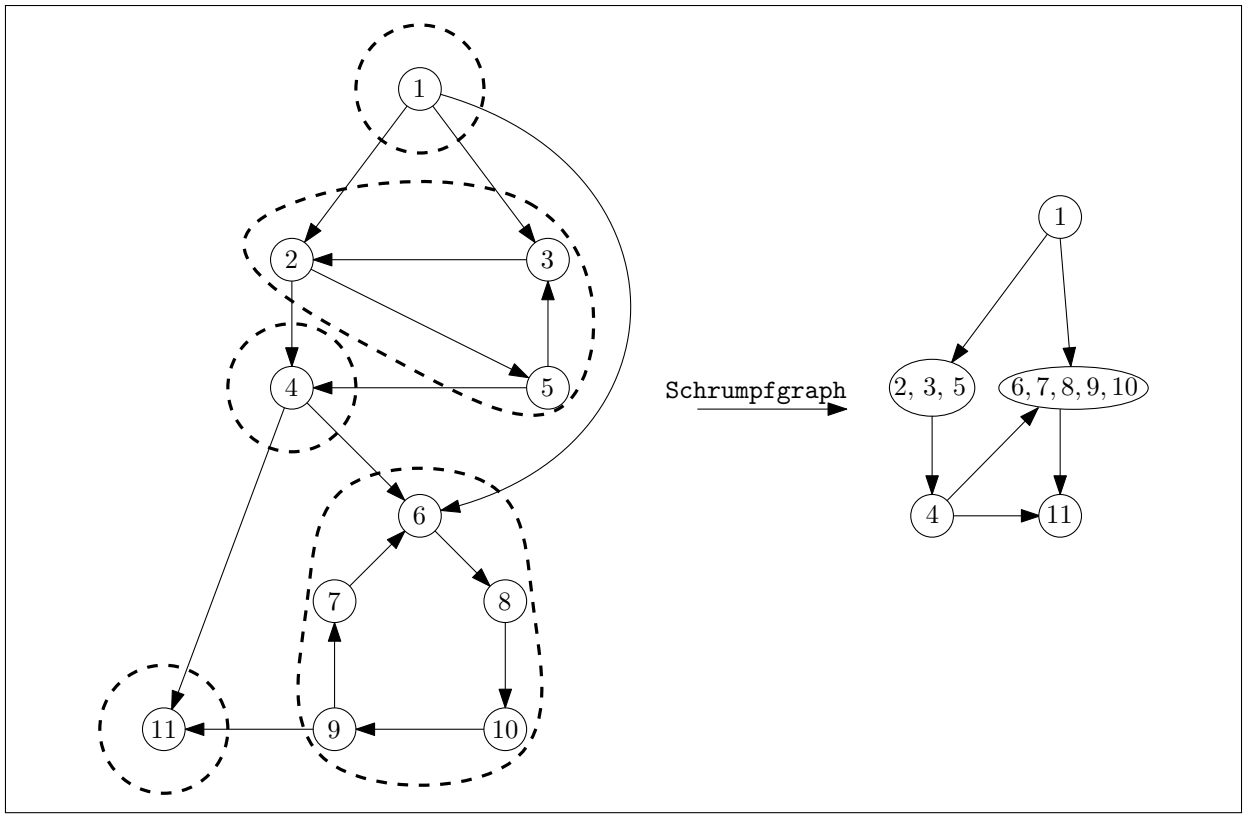
b. Sei die parallele Ausführungszeit eines Algorithmus gegeben durch $T_{par}(n,p) = \frac{n}{\log(2p)}$. Die Laufzeit des besten sequentiellen Algorithmus betrage $T_{seq} = n$. Geben Sie die Effizienz E und die Arbeit W des parallelen Algorithmus an. [2 Punkte]

Lösung

$E = \frac{n/\log(2p)}{p \cdot n} = \frac{\log(2p)}{p}$
$W = p \cdot \frac{n}{\log(2p)}$

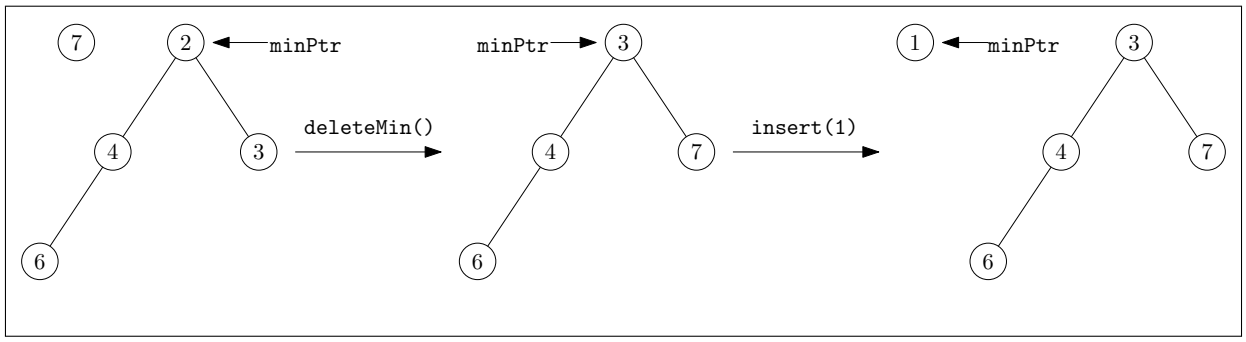
c. Markieren Sie im unten stehenden Graphen alle starken Zusammenhangskomponenten **und** zeichnen Sie den dazugehörigen Schrumpfgraphen.
(Zwei Kopien. Falls Sie mehr als eine der Kopien verwenden, markieren Sie deutlich diejenige, die bewertet werden soll. Andernfalls wird die Aufgabe mit 0 Punkten bewertet.) [2 Punkte]

Lösung



d. Gegeben sei ein Fibonacci Heap im unten stehenden Zustand. Geben Sie jeweils den Zustand nach einer `deleteMin`-Operation und einer darauf folgenden `insert`(1)-Operation an. [2 Punkte]

Lösung



e. Betrachten Sie das Ski Rental Problem aus der Vorlesung mit Mietpreis 2 und Kaufpreis 10. Sie verwenden folgenden Algorithmus *ALG*:
Sie beginnen damit die Ski zu mieten. Wenn Sie an Tag 4 noch fahren werden, kaufen Sie mit 50% Wahrscheinlichkeit. Falls Sie an Tag 4 nicht gekauft haben und an Tag 5 auch noch fahren werden, kaufen sie.
Berechnen Sie den erwarteten Competitive Ratio $\max_I \mathbb{E}[\frac{ALG(I)}{OPT(I)}]$. [4 Punkte]

Lösung

Angenommen es wäre optimal zu kaufen, wir also mindestens 5 Tage fahren. Unsere erwarteten Kosten sind $(0.5 \cdot 6 + 0.5 \cdot 8) + 10 = 17$. Optimal wären 10. $\frac{17}{10} = 1.7$.
Wenn wir nur 4 Tage fahren, haben wir erwartete Kosten von $0.5 \cdot (10+6) + 0.5 \cdot 8 = 12$. Optimal wären 8. $\frac{12}{8} = 1.5$.
Wenn wir weniger als 4 Tage fahren, sind wir optimal.
Die erwartete Competitive Ratio ist also 1.7, was besser ist als die Competitive Ratio des besten deterministischen Algorithmus (1.8).

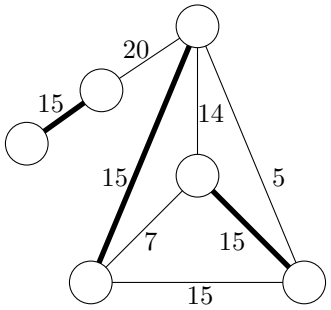
Lösungsvorschlag

Aufgabe 2. Maximum Weight Matchings [9 Punkte]

Gegeben sei ein ungerichteter, zusammenhängender Graph $G = (V, E)$ mit Kantengewichtsfunktion $w : E \to \mathbb{R}^+$. Ein *Matching* ist eine Teilmenge M der Kanten in G, sodass keine zwei Kanten in M einen Knoten gemeinsam haben. Ein *Maximum Weight Matching* ist ein Matching mit maximalem Gewicht $w(M) = \sum_{e \in M} w(e)$ unter allen Matchings.

a. Geben Sie für den unten stehenden Graphen ein Maximum Weight Matching an, indem Sie die entsprechenden Kanten hervorheben.
(Zwei Kopien. Falls Sie mehr als eine der Kopien verwenden, markieren Sie deutlich diejenige, die bewertet werden soll. Andernfalls wird die Aufgabe mit 0 Punkten bewertet.) [1 Punkte]

Lösung



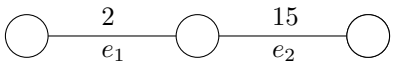
b. Findet der folgende Algorithmus eine 3-Approximation eines Maximum Weight Matchings? Falls ja, beweisen Sie. Falls nein, geben Sie ein Gegenbeispiel an, und nennen Sie das vom Algorithmus gefundene Matching und ein Maximum Weight Matching. Gehen Sie davon aus, dass beim Durchlaufen der Kanten jede Reihenfolge möglich ist. [3 Punkte]

Algorithmus 1 `MaximumMatchingApproximation`

$M \leftarrow \emptyset$
for $e = \{u, v\} \in E$ **do**
  if u und v sind nicht markiert **then**
    $M \leftarrow M \cup \{e\}$
    markiere u und v
return M

Lösung

Der Algorithmus berechnet keine 3-Approximation. Im unten stehenden Beispiel, ist das optimale Matching $M^* = \{e_2\}$. Falls der Algorithmus allerdings zuerst e_1 betrachtet, findet er das Matching $M = \{e_1\}$ und damit $w(M) = 2 < 5 = w(M^*)/3$.



c. Geben Sie einen Algorithmus mit $O(|E|\log|E|)$ Laufzeit an, der eine 2-Approximation eines Maximum Weight Matchings berechnet.
Beweisen Sie außerdem, dass ihr Algorithmus eine 2-Approximation berechnet. [5 Punkte]

Lösung

Algorithmus 2 `MaximumMatchingApproximation`

$M \leftarrow \emptyset$
for $e = \{u, v\} \in E$ in absteigender Reihenfolge nach Gewicht **do**
  if u und v sind nicht markiert **then**
    $M \leftarrow M \cup \{e\}$
    markiere u und v
return M

Wir betrachten ein Maximum Weight Matching M^* und beobachten, wodurch der Algorithmus dazu führen kann, dass Kanten aus M^* nicht in der Lösung enthalten sind. Dazu formulieren wir den Algorithmus so um, dass immer, wenn eine Kante e zu M hinzugefügt wird, e und alle inzidenten Kanten aus E entfernt werden, da diese nicht mehr im Matching enthalten sein können. Der Algorithmus wird dann solange ausgeführt, bis $E = \emptyset$. Wenn nun eine Kante e zu M hinzugefügt wird, unterscheiden wir zwei Fälle:

1. $e \in M^*$: Dann ist nichts weiter zu tun.
2. $e \notin M^*$: Nun können maximal zwei Kanten $e_1, e_2 \in M^*$ aus E entfernt werden. Da e vor e_1 und e_2 betrachtet wurde, ist $w(e_1), w(e_2) \leq w(e)$ und somit auch $w(e_1) + w(e_2) \leq 2 \cdot w(e)$.

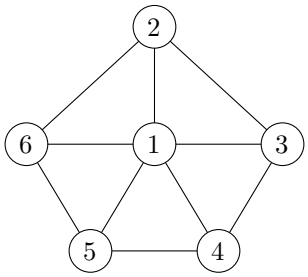
Da am Ende E leer ist, wurden alle Kanten aus M^* entweder zu M hinzugefügt oder aus E entfernt. Da jede Kanten aus M entweder auch in M^* ist oder nach Fall 2 maximal zwei Kanten aus M^* mit kleinerem Gewicht "verhindert" hat, ist $w(M) \geq w(M^*)/2$.

Lösungsvorschlag

Aufgabe 3. Zufallsgraphen [8 Punkte]

Betrachten Sie folgende Familie von Graphen mit $n > 4$ Knoten: $n-1$ Knoten bilden einen Kreis. Ein letzter Knoten ist mit allen anderen Knoten verbunden. Sonst enthält der Graph keine weiteren Kanten. Im Folgenden werden wir einen solchen Graphen einen *Kreisstern* nennen.

a. Berechnen Sie die Wahrscheinlichkeit, dass im Fall $n = 6$ der unten stehende Kreisstern durch das $G(n,p)$-Zufallsgraphenmodell nach Erdős–Rényi erzeugt wird. [2 Punkte]



Lösung

In einem Graphen mit 6 Knoten gibt es maximal 15 Kanten. Der gegebene Kreisstern hat 10 Kanten. Damit ist die Wahrscheinlichkeit für diesen Graphen $p^{10} \cdot (1-p)^5$.

b. Wie viele mögliche Kreissterne gibt es auf der Knotenmenge $V = \{1,\ldots,n\}$? Begründen Sie Ihre Antwort. [3 Punkte]

Lösung

Jeder Knoten kann Mittelpunkt sein $\Rightarrow n$. Nun muss noch betrachtet werden, wie viele mögliche Kreise mit $n-1$ Knoten es gibt. Wir können die Kreise als Permutationen der Zahlen $1,\ldots,n-1$ interpretieren. Davon gibt es $(n-1)!$. Da aber alle "Shifts" von Permutationen identische Kreise erzeugen ($(1,2,3)$ ist der gleiche Kreis, wie $(2,3,1)$), müssen wir noch durch $n-1$ teilen. Außerdem entsprechen auch zwei Permutationen, von denen eine die Umkehrung der anderen ist, den gleichen Kreisen, also muss noch durch 2 geteilt werden. Zusammen ergibt sich $n \cdot \frac{(n-1)!}{2(n-1)} = \frac{n!}{2(n-1)}$.

c. Wie hoch ist die Wahrscheinlichkeit, dass nach dem $G(n,p)$-Zufallsgraphenmodell ein Kreisstern erzeugt wird? [3 Punkte]

Lösung

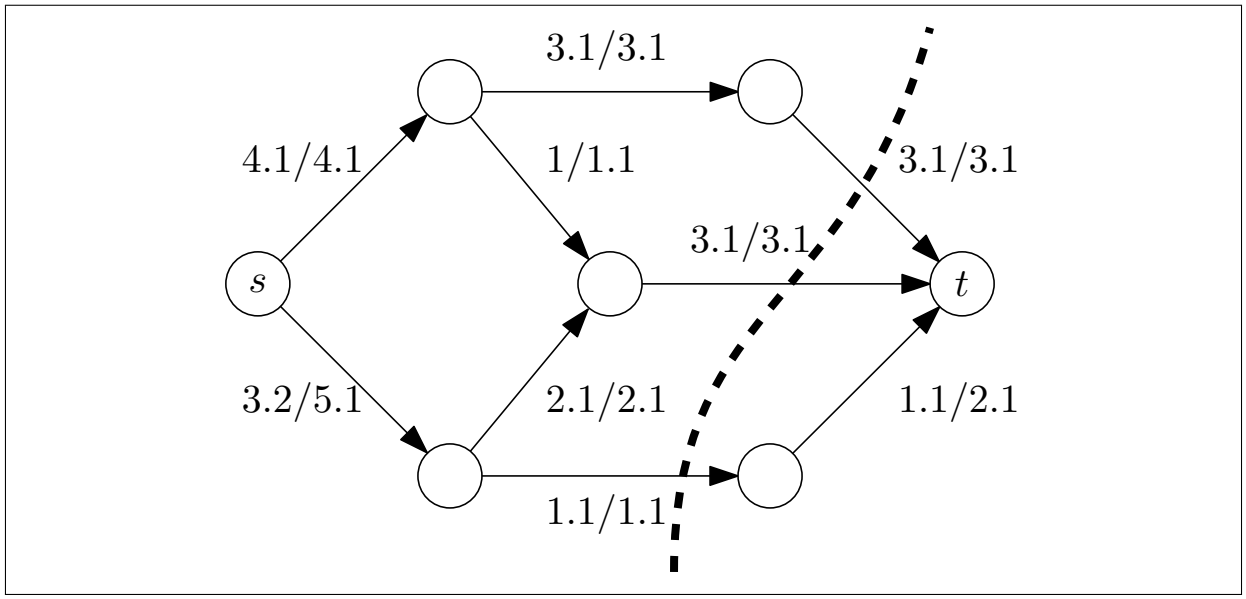
Wir berechnen zunächst die Anzahl Kanten in einem Kreisstern mit n Knoten. Der mittlere Knoten hat $n-1$ Kanten. Ein Kreis mit $n-1$ Knoten besteht ebenfalls aus $n-1$ Kanten. Ein Kreisstern hat also $2(n-1)$ Kanten. Da alle Graphen mit m Kanten gleich wahrscheinlich sind, ergibt sich mit der Anzahl möglicher Kreissterne aus der letzten Teilaufgabe eine Wahrscheinlichkeit von $\frac{n!}{2(n-1)} \cdot p^{2(n-1)} \cdot (1-p)^{\binom{n}{2}-2(n-1)}$.

Lösungsvorschlag

Aufgabe 4. Flussalgorithmen [11 Punkte]

a. Finden Sie in folgendem Flussnetzwerk den maximalen Fluss, indem sie die Kanten mit dem Fluss über die jeweilige Kante beschriften. Die Zahlen an den Kanten geben ihre Kapazität an. [1 Punkte]

Lösung



b. Beweisen Sie, dass Ihr Fluss aus der vorhergehenden Teilaufgabe maximal ist. [2 Punkt]

Lösung

Der eingetragene Fluss hat Wert 7.3. Der oben eingezeichnete Schnitt hat ebenfalls Wert 7.3. Nach dem Min-Cut-Max-Flow-Theorem ist der angegebene Fluss daher maximal.

c. Betrachten Sie nun ein Flussnetzwerk F' mit Kapazitäten c', das aus einem Flussnetzwerk F mit natürlichzahligen Kapazitäten c konstruiert wird, indem alle Kapazitäten um einen Wert $\alpha > 0$ erhöht werden, sodass die Kapazität jedes Schnittes um weniger als 1 erhöht wird. (Also $c'(e) = c(e) + \alpha$ für alle Kanten e)

1. Finden Sie ein solches $\alpha > 0$ und beweisen Sie, dass die Kapazität jedes Schnittes um weniger als 1 steigt.

   Hinweis: Es genügt, einfache Graphkennzahlen zu verwenden.

2. Sei M die Menge der minimalen $s-t$-Schnitte in F und M' die Menge der minimalen $s-t$-Schnitte in F'. Beweisen Sie, dass gilt: $M' \subseteq M$.

[4 Punkte]

Lösung

1. $\alpha = \frac{1}{m+1}$ erfüllt die gewünschte Eigenschaft, wobei m die Anzahl der Kanten in F ist. Da jeder Schnitt aus maximal m Kanten besteht, wird die Kapazität jedes Schnittes um maximal $m \cdot \alpha = \frac{m}{m+1} < 1$ erhöht.

2. Sei C die Menge aller $s-t$-Schnitte in F. Wir zeigen, dass für alle $s-t$-Schnitte $l \in C \setminus M$ gilt: $l \notin M'$ und damit $M' \subseteq M$. Dafür ist wichtig zu sehen, dass durch die natürlichzahligen Kapazitäten die Kapazität eines nicht-minimalen Schnittes um mindestens 1 größer ist als der eines minimalen Schnittes. Wir betrachten $l \in C \setminus M, m \in M$: Nach Definition von c' gilt: $c'(m) < c(m) + 1 \leq c(l) < c'(l)$ und damit auf jeden Fall $l \notin M'$.

d. Gegeben sei ein Flussnetzwerk F mit Start- und Zielknoten s und t, bei dem die Kapazitäten aller Kanten natürliche Zahlen sind. Geben Sie einen Algorithmus an, der mittels Flussalgorithmen einen minimalen $s-t$-Schnitt mit minimaler Anzahl Kanten findet. [2 Punkte]

Lösung

Konstruiere F', wie oben. Finde darin einen maximalen $s-t$-Fluss f. Der gesuchte Schnitt ist dann $(S, V \setminus S)$, wobei S die Menge der Knoten ist, die in G_f von s aus erreichbar sind.

e. Beweisen Sie die Korrektheit Ihres Algorithmus aus der vorherigen Teilaufgabe. [2 Punkte]

Lösung

Wie oben gezeigt wurde, sind die minimalen $s-t$-Schnitte in F' auch minimale $s-t$-Schnitte in F. Da der Wert jedes $s-t$-Schnittes linear mit der Anzahl Kanten im Schnitt gewachsen ist, ist der minimale Schnitt in F' ein minimaler Schnitt mit minimaler Anzahl Kanten in F.
Wie in der Vorlesung gezeigt wurde, liefert die angegebene Konstruktion von S einen minimalen Schnitt in F'.

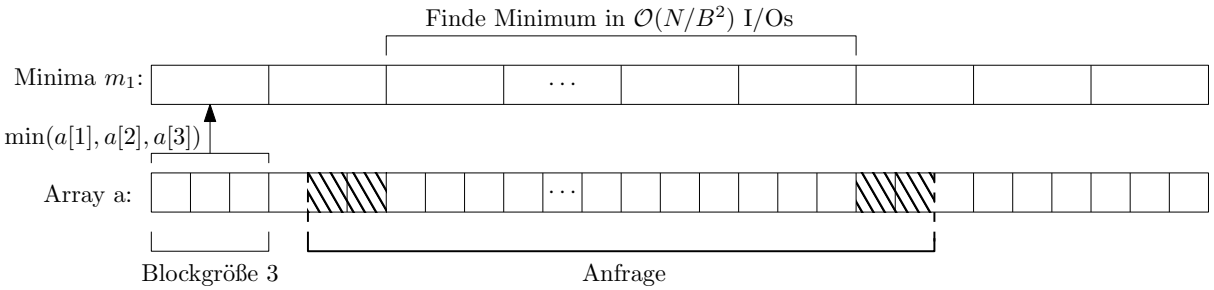
Lösungsvorschlag

Aufgabe 5. External Memory RMQ [12 Punkte]

Gegeben sei ein Array im externen Speicher. Sei N die Größe des Arrays und B die Größe der Blöcke, die in einer I/O (Ein- oder Ausgabe)-Operation vom externen Speicher gelesen bzw. in den externen Speicher geschrieben werden können. Wir möchten auf diesem Array im externen Speicher Range Minimum Queries (RMQs) beantworten. Gehen Sie für alle Aufgabenteile davon aus, dass N ein Vielfaches von B ist. Des Weiteren sei $B > \log_2 N$.

a. Zeigen Sie, wie Sie durch geeignete Vorberechnungen mit $O(\frac{N}{B})$ I/O-Operationen RMQs auf beliebigen Intervallen $[i, j]$ in $O(\lceil \frac{j-i}{B^2} \rceil)$ I/O-Operationen beantworten können. Analysieren Sie die Anzahl I/O-Operationen für Anfragen und Vorberechnungen.
Hinweis: Teilen Sie das Array in N/B Blöcke auf. [5 Punkte]

Lösung



Wir berechnen für jeden Block der Größe B den minimalen Wert und speichern ihn in einem neuen Array m_1 der Größe $\frac{N}{B}$ im externen Speicher. Dafür muss ein Mal das ganze Eingabearray gescant werden, was $O(\frac{N}{B})$ I/O-Operationen benötigt. Um eine Anfrage auf dem Intervall $[i, j]$ zu beantworten, finden wir im Bereich $[i', j']$, mit $i' = \lceil \frac{i}{B} \rceil + 1, j' = \lfloor \frac{j}{B} \rfloor$, das Range Minimum durch einen Scan auf m_1 in $O(\frac{j'-i'}{B}) = O(\frac{j-i}{B^2})$ I/O-Operationen. Dann müssen noch die durch die obigen Rundung von i und j abgeschnittenen Bereiche betrachtet werden. Da auf beiden Seiten der Range weniger als ein Block abgeschnitten wurde, werden dafür nur 2 weitere I/O-Operationen benötigt.

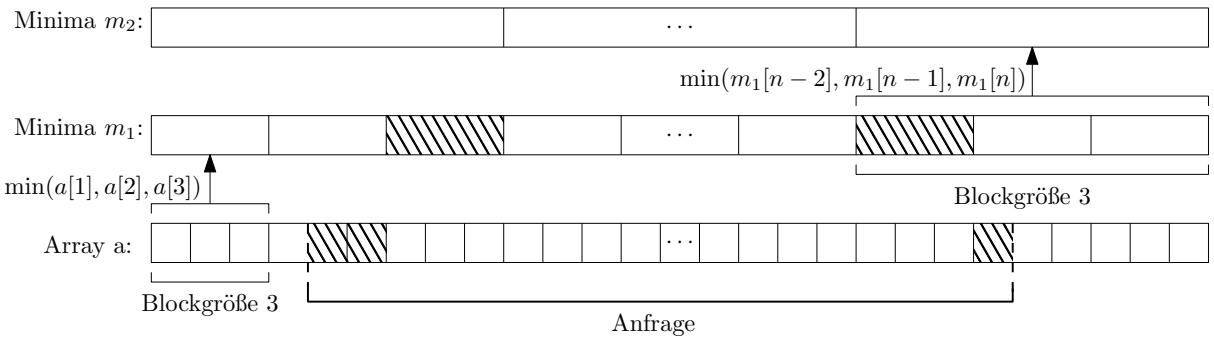
b. Gegeben sei eine RMQ-Datenstruktur, deren Aufbau für n Elemente im internen Speicher $O(n \log n)$ Zeit benötigt und mit der beliebige RMQ-Anfragen mit $O(1)$ beantwortet werden können. Nutzen Sie diese um in $O(N)$ I/O-Operationen eine Datenstruktur aufzubauen, mit der beliebige RMQs auf dem Array im externen Speicher mit $O(1)$ I/O-Operationen beantwortet werden können. Analysieren Sie die Anzahl I/O-Operationen für Anfragen und Vorberechnungen.
Hinweis: Ein Algorithmus, der im internen Speicher $O(f(n))$ Zeit benötigt, kann im externen Speichermodell mit $O(f(N))$ I/O-Operationen implementiert werden. [3 Punkte]

Lösung

Wir wenden die gegebene Datenstruktur auf m_1 aus der vorherigen Teilaufgabe an. Dafür werden $O(\frac{N}{B} \log \frac{N}{B}) < O(\frac{N}{B} \log N) < O(N)$ I/O-Operationen benötigt. Die Anfrage wird dann wie oben durchgeführt mit der Unterscheidung, dass die Anfrage auf m_1 mit Hilfe der Datenstruktur in $O(1)$ I/O-Operationen durchgeführt wird.

c. Verbessern Sie Ihre Datenstruktur so, dass die Vorberechnung nur noch $O(\frac{N}{B})$ I/O-Operationen benötigt (und Anfragen immer noch in $O(1)$ I/O-Operationen beantwortet werden können). Beschreiben Sie Ihre Datenstruktur und wie damit Anfragen beantwortet können. Analysieren Sie außerdem die Anzahl I/O-Operationen für Vorberechnung und Anfragen.
Hinweis: Teilen Sie die Blöcke aus der ersten Teilaufgabe wiederum in Blöcke auf, sodass die Anzahl Elemente, die ihre Datenstruktur verwalten muss, nur noch in $O(\frac{N}{B^2})$ liegt. [4 Punkte]

Lösung

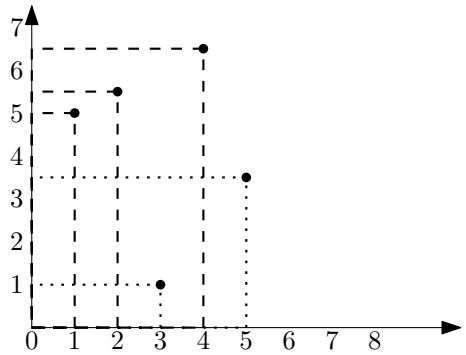


Wir teilen m_1 in Blöcke der Größe B ein und konstruieren ein zweites Hilfsarray m_2, in dem wir die Minima dieser Blöcke speichern. Wir verwenden die Datenstruktur aus der letzten Teilaufgabe dann für m_2 statt m_1. Diese kann in $O(\frac{N}{B^2}\log\frac{N}{B^2}) < O(\frac{N}{B^2}\log N) < O(\frac{N}{B^2}B) = O(\frac{N}{B})$ I/O-Operationen vorberechnet werden. Wir berechnen wieder $i' = \lceil\frac{i}{B}\rceil + 1, j' = \lfloor\frac{j}{B}\rfloor$ für Anfragen auf m_1. Für Anfragen auf m_2 berechnen wir $i'' = \lceil\frac{i'}{B}\rceil + 1, j'' = \lfloor\frac{j'}{B}\rfloor$. Die Anfrage auf m_2 kann mittels der Datenstruktur in $O(1)$ I/O-Operationen beantwortet werden. Der Rest der Anfrage auf m_1 kann wie in der ersten Teilaufgabe mit 2 zusätzlichen I/O-Operationen beantwortet werden. Dann bleiben noch wie in der ersten Teilaufgabe die 2 I/O-Operationen auf dem Eingabearray.

Lösungsvorschlag

Aufgabe 6. Dominierende Rechtecke [7 Punkte]

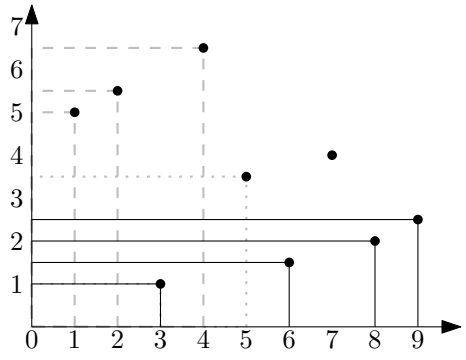
Gegeben sei eine Menge von Punkten $S = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$, mit $x_i, y_i \in \mathbb{R}^+$ für alle i, die wir als Rechtecke vom Ursprung zum jeweiligen Punkt interpretieren. Ein Rechteck (x_i,y_i) *dominiert* ein anderes Rechteck (x_j,y_j), wenn $x_i > x_j$ und $y_i > y_j$.
Wir suchen die Länge k der längsten Folge von Rechtecken $\langle R_1,\ldots,R_k\rangle$ mit $R_i \in S$, sodass gilt: $\forall i > j : R_i$ dominiert R_j.
Beispiel: In der Abbildung sind zwei Folgen dominierender Rechtecke der Größe 2 und 3 zu sehen: Die gepunktet eingezeichneten und die gestrichelt eingezeichneten (Nur zwei Beispiele - es gibt noch weitere solche Mengen). Der gesuchte Wert k ist hier 3.



a. Im unten stehenden Bild wurden dem Beispiel einige Punkte hinzugefügt. Geben Sie den gesuchten Wert k an. [1 Punkt]

Lösung

Die eingezeichneten Rechtecke erfüllen die gewünschte Eigenschaft. Der gesuchte Wert k ist hier also 4.



b. Angenommen, Sie haben bereits alle Punkte mit $x \leq 7$ betrachtet. Sie wissen, dass es dort eine dominierende Folge von Rechtecken der Größe 1 gibt, bei der das äußerste Rechteck den y-Wert 1,5 besitzt. Außerdem wissen Sie, dass es eine Folge von dominierenden Rechtecken der Größe 2 gibt, bei der das äußere Rechteck den y-Wert 2,5 besitzt, eine der Größe 3 mit y-Wert 3 und eine der Größe 4 mit y-Wert 8. Die unten stehende Tabelle wiederholt diese Werte.

1. Sie betrachten nun einen zusätzlichen Punkt $(8,5)$. Was ist der gesuchte Wert k?
2. Was ist der gesuchte Wert k, wenn ein weiterer Punkt $(9,6)$ hinzukommt?

[2 Punkte]

k	y
1	1,5
2	2,5
3	3
4	8

Lösung

1. Da $8 > 7$ und $5 > 3$, umschließt $(8,5)$ das äußerste Rechteck in der Folge der Länge 3. k ist hier also 4.
2. Dieses wiederum wird von $(9,6)$ umschlossen. Daher ist $k = 5$.

c. Entwerfen Sie einen Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$ in Pseudocode, der eine Menge von Punkten S erhält und den Wert k aus der Aufgabenbeschreibung zurückgibt. Sie können davon ausgehen, dass die x- und y-Koordinaten jeweils paarweise verschieden sind.

Lösung

Algorithmus 3 `DominatingRectangles`

$k \leftarrow 0$
$R \leftarrow [0, \infty, \infty, \ldots, \infty]$(Array der Größe n)
Sortiere S nach seiner x-Koordinate
for i from 1 to n **do**
    Finde größtes j mit $R[j] < S[i].y$ mittels binärer Suche
    $R[j+1] \leftarrow S[i].y$
    $k \leftarrow \max\{j+1, k\}$
return k

[4 Punkte]