Sortieren/Auswahl

Sortieralgos
Bogosort: alle $\pi \in S_n$ prüfen
$w: n!(n-1)\ avg: \frac{n!}{2}(\Omega(n!))$
Selection-Sort: n-mal Min. ausgeben
$w: \Theta(n^2)$
Merge-Sort: $\frac{1}{2}$ sortieren, mergen.
$\forall: \Theta(n \log(n))$
Quicksort: Nach Pivotelement partitionieren und sort.
$w: \Theta(n^2)\ avg: \Theta(n \log n)$
Suchproblem ist $\Omega(n \log n)$, bei $|\mathcal{U}| < \infty$ nicht.
Countingsort: Zählen der Elemente aus $\mathcal{U}$
BucketSort: $f: \mathcal{U} \to [r]$ nutzen, dann Countingsort
Radixsort: Buckesort auf Buchstaben mehrerer Wörter von hinten aufgerollt

Select
Algorithmus: Gesucht: k-kleinstes Element in sort. Liste
Zerlege über Pivotelement und partioniere wie bei Quicksort
Pivotelement: Zufällig wählen
$avg: \mathcal{O}(n)$
Deterministisch wählen $\Theta(n)$

- Zerlege alles in 5-er Blöcke
- Sortiere die Mediane und bestimme Median rekursiv

Suchen
a in einer sort. Folge S suchen
Binärsuche: $k = \lfloor \frac{l+r}{2} \rfloor$
Interpolsearch: $\mathcal{U} = [0, 1]$,
$S_0 := 0, S_{n+1} = 1$
$k = l - 1 + \lfloor \frac{a - S_{l-1}}{S_{r+1} - S_{l-1}}(r - l + 1) \rfloor$
$w: \Theta(n)\ avg: \mathcal{O}(\log \log n)$.
Quadbinsearch: Sprünge von $k \pm i(\lfloor \sqrt{r - l + 1} \rfloor + 1)$, suche im verbleibendem Intervall.
$avg: \mathcal{O}(\log \log n)$

Regmaschinen

Programme
A := B [op C], goto L,
GZ, GGZ, GLZ B,L. $(R_i) = R_{R_i}$
EKM: 1 Schritt/Register 1 Einheit
LKM: $L(R) = \log_2(|R|) + 1$ als Kosten/Register

Sätze
$T(n)$ LKM auf RAM, Turingmaschine in $\mathcal{O}(T(n)^5)$.
RAM und TM polynomiell verwandt.

Datenstrukturen

Hashing
$h: U \to [m]$, $m \in \Theta(|S|)$ ist sinnvoll. Wenn h gut verteilt, dann alles in $\mathcal{O}(1)$.

Bäume
Sind maximal azyklische, minimal zusammenhängende Graphen. Eine Kante mehr und ein Zyklus entsteht.
Binärbäume: Elemente im linken/rechten Teilbaum sind kleiner/größer als Wurzel des Teilbaums. Erwartete Höhe in $\mathcal{O}(\log n)$. n-mal Einfügen in $avg: \Theta(n \log n)$.
Tries:

- Verzweigungen sind Buchstaben
- Knoten sind (Teil-)Wörter, markiert falls Wörter

AVL-Bäume
Erwartete Höhe bei n inneren Knoten $\Theta(\log n)$
$|h(T_l(v)) - h(T_r(v))| \leq 1\ \forall v$, sonst Balancieren mit Einfach/Doppeltrot.

(a, b)-Bäume
$b \geq 2a - 1, a > 2$
$\forall v$ innere Knoten: $|c(v)| \in [a, b]$
$|c(root(T))| \geq 2$, Daten in Blättern
$\log_b(n) \leq h \leq \log_a(n) + 1$
Alle Blätter haben gleiche Tiefe
Der kleinste ist (2,3)
EINF $\in \mathcal{O}(\log n)$ (evt. Spalten!)
STREICH $\in \mathcal{O}(\log n)$ (adoptieren/verschmelzen)
B-Bäume sind $(k, 2k-1)$ Bäume

RS-Bäume

- Rote haben 2 schwarze Kinder
- $\forall$ Pfade Wurzel $\to$ Blatt enthalten gleich viele schwarze
- Jedes Blatt ist schwarz
- $(2, 4)$-Bäume wenn rote mit schwarzen Vätern verschmolzen

BB[α] Bäume
Balance$(v) \in [\alpha, 1 - \alpha]$ für innere Knoten v, *Balance*$(v) = \frac{|T_l(v)|}{|T(v)|}$
(Anz. Blätter!)
Balancieren wie bei AVL.

Bruderbäume

- innere Knoten v: $|c(v)| \in [1, 2]$
- Knoten mit 1 Kind hat Bruder mit 2 Kindern
- $\Rightarrow$ jeder Knoten hat mind. 2 Enkel

Höhe in $\mathcal{O}(\log(n))$. Verschmilzt man Väter mit ihren Einzelkindern, dann hat man einen AVL-Baum.

Optimale Suchbäume
$a_1 < \ldots < a_n$ mit p_i als Auftrittswkeit gegeben, q_i W-keit für (a_i, a_{i+1}), $a_0 = -\infty, a_{n+1} = \infty$
Konstruktion
$\in \mathcal{O}(n^3)(\Theta(n^2) = space)$
$r_{i,j}$: Index der Wurzel für $T_{i,j}$
$c_{i,j}$: Kosten von $T_{i,j} = w_{i,j} P_{T_{i,j}}$
$w_{i,j}$: $P(a \in [a_i, a_j])$
for i = 0..n
$w_{i+1,i} = q_i$
$c_{i+1,i} = 0$
endfor
for k = 0..n-1, i = 1..n-k
$j = i + k$
Sei $m \in [i, j]$ mit $c_{i,m-1} + c_{m+1,j}$ min.
$r_{i,j} = m$
$w_{i,j} = w_{i,m-1} + w_{m+1,j} + p_m$
$c_{i,j} = c_{i,m-1} + c_{m+1,j} + w_{i,j}$
endfor
Tabelle bauen mit Spalten $i = 0..n$ und Spalten init und $k = 0..n-1$.
Index Wurzel ist $m_w = r_{1,n}$.
$m_l = r_{1,m_w-1}, m_r = r_{m_w+1,n}$.
Verbesserung Wurzelsuche: $\mathcal{O}(n^2)$

Union-Find
$S = [n], \mathcal{S} = \bigcup_{i=1}^k S_i$ Partition.
Vereinige(S_i, S_j):
$\mathcal{S} := \mathcal{S} \backslash \{S_i, S_j\} \cup \{S_i \cup S_j\}$
$(\mathcal{O}(1) =)$
Finde(a): finde $a \in S_i \in \mathcal{S}$. $(\Theta(n))$.
Strukturell als Wald, jedes S_i ist ein Baum, die Elemente aus S_i zeigen auf ihren Vaterknoten mit einen Repräsentanten von S_i als Wurzel.
Höhenbalancierung: Finde $\in \mathcal{O}(\log n)$
Pfadkompression: (Aufsammeln der Knoten des Suchpfads) $m\times$ Finde $\in \mathcal{O}((m + n) \log^*(n))(avg)$, auch in $\mathcal{O}(m\alpha(m, n))$ mit
$\alpha(m, n) = \min\{z \geq 1, A(z, 4 \cdot \lceil \frac{m}{n} \rceil) > \log n\}$

Graphen

Definitionen
$G = (V, E)$ als Graph gegeben.
$|V| = n, |E| = m$.
$W = (v_1, \ldots, v_n), v_i \in V$ heißt **Weg**. Wenn v_i paarweise ungleich, dann ist er **einfach**. Wenn $v_1 = v_n$ dann ist es ein **Kreis**.
Der **Durchmesser** von G ist das Maximum der Abstände zwischen den Knotenpaaren.
Ein Graph ist **zusammenhängend**, wenn ein Weg zwischen jedem Knotenpaar existiert. Bei gerichteten Graphen **stark zusammenhängend** wenn es für beide Richtungen Wege gibt.
Eine **Zusammenhangskomponente** ist ein maximaler Teilgraph von G, der zusammenhängend aber nicht erweiterbar ist.
Die **Inzidenzmatrix** ist eine $n \times m$ Matrix, in der Spalte jeder Kante (v_i, v_j) wird in Zeile $i|j$ eine $1| - 1$ eingetragen, wenn G ungerichtet, dann Betrag der Matrix nehmen.
Ein Graph ist **Bipartit**, wenn es eine 2-Partition gibt, sodass nur Kanten zwischen den Partitionsmengen verlaufen.
Teilgraph: $V', E' \subseteq V, E'$, **induziert** wenn alle Kanten übernommen werden.

Traversierung
Breitensuche: Suche über Prioritätswarteschlange (Schwestern dann Kinder), findet immer kürzesten Weg *Tiefensuche:* Rekursiv alle Kinder durchsuchen. (Findet Zusammenhangskomp.)
Beide in $\mathcal{O}(nm)$.

Topologisches Sort
Ordnet bei gerichteten, azyklischen Graphen, die Knoten so an, dass die Kanten von links nach rechts gehen. Zum bestimmen über Tiefensuche nach Abfertigungszeit sortieren.

Zusammenhangskomp.
Führe $DFS(G)$ aus, nummeriere in DFS-Reihenfolge. Höchste Nummern jeweils in $DFS(G^T)$ wählen. Die Tiefensuchbäume sind nun Zusammenhangskomponenten.

MSTs
Lemma: Spannender Wald (V_i, T_i), wenn $e = (u \in V_1, v \notin V_1)$ mit min. Kosten, dann $\exists$ MST von G, der $T \cup \{e\}$ enthält.
Kruskal $\in \mathcal{O}(m \log m)$
Fange mit den Bäumen $\{v_i\}$ an. Finde minimale Kante, finde Bäume davon, wenn ungleich Vereinige sie. Entferne Kante.

Single-shortest-Paths
Dijkstra $\in \mathcal{O}((m + n) \log n)$
$S := \{s\} D[s] := 0$
forall $V \ni v \neq s$, $D[v] := C(s, v)$
while $V - S \neq \emptyset$
$w := \min_{w \in V-S} D[w]$
$S \cup = w$
for each $u \in V - S$, $(w, u) \in E$
$D[u] := \min(D[u], D[w] + C(w, u))$
endwhile

All-shortest-Paths
Floyd-Warshall $\in \Theta(n^3)$
for $i, j = 1..n$
$d_{i,j}^{(0)} = \begin{cases} c(i,j), & (i,j) \in E \\ \infty, & else \end{cases}$
for $k, i, j = 1..n$
$d_{i,j}^{(k)} = \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)})$

Flüsse
Netz $(G, c: E \to \mathbb{R}_{\geq 0}, s, t)$. Fluss $f: V^2 \to \mathbb{R}$:

- $f(u, v) \leq c(u, v)\ \forall u, v \in V$
- $f(u, v) = f(v, u)\ \forall u, v \in V$
- $\sum_{v \in V} f(u, v) = 0\ \forall u \in V \backslash \{s, t\}$

$|f| = \sum_{v \in V}(f(s, v))$
$X, Y \subseteq V: f(X, Y) = \sum_{x \in X, y \in Y} f(x, y)$
Es gilt (Lemma) $(X, Y, Z \subseteq V)$

- $f(X, X) = 0$
- $f(X, Y) = -f(Y, X)$
- f ist $\cup$-bilinear unter disjunkten Mengen

Min-Cut-Max-Flow
Für (G, c, s, t) und f sind äquivalent

- f ist maximaler Fluss
- Es ex. keine aug. Wege in G_f
- Es ex. ein Cut (S, T), sodass $c(S, T) = |f|$

Ford-Fulkerson $\in \mathcal{O}(|f^*|m)$
while $\exists$ augm. Weg $s \to t$ P in G_f erhöhe Fluss für jede Kante im Weg

Edmonds-Karp $\in \mathcal{O}(nm^2)$
Breitensuche statt Tiefensuche, sonst wie FF. Augmentierungen $\in \mathcal{O}(nm)$.

Bipartites Matching
Ein **Matching** $M \subseteq E$, wenn keine zwei Kanten gleiche Endpunkte haben. Finden mit Superquellen/senken Trick.
Satz: $c(E) \subseteq \mathbb{N}_0$, dann $|f^*| \in \mathbb{N}_0$.
Bester Algo $\in \mathcal{O}(\sqrt{n}m)$.

String-Pattern-Matching

Endliches Alphabet Σ gegeben, $T \in \Sigma^n$, $P \in \Sigma^m$, $m \leq n$. Wir suchen die Vorkommen von P in T.
Präfix: w ist ein Anfangsstück (Präfix) von x ($x \sqsubset w$), wenn $\exists u \in \Sigma^*: x = wu$ mit $w, x \in \Sigma^*$.
Suffix: w ist ein Endstück (Suffix) von x ($x \sqsupset w$), wenn $\exists u \in \Sigma^*: x = uw$, mit $w, x \in \Sigma^*$.
Der naive Algorithmus prüft einfach alle möglichen Anfangstellen von $1..n - m$ von $T \in \Theta(nm)$.

Knuth-Morris-Pratt

Algorithm 1 KMP-Algo $\in \mathcal{O}(n + m)$

Berechne für P die Präfixfunktion π
$q := 0$;
for $i := 1, \ldots, n$ **do**
  while $((q > 0) \wedge (P[q + 1] \neq T[i]))$ **do**
    $q := \pi[q]$
  end while
  if $(P[q + 1] = T[i])$ **then**
    $q := q + 1$;
  end if
  if $(q = m)$ **then**
    Match$(i - m)$;
    $q := \pi[q]$
  end if
end for

Suffixbäume

- n Blätter
- innere Knoten (außer Wurzel) hat mind. zwei Kinder
- Kanten sind mit nichtleeren Teilwörtern von T beschriftet
- Zwei verschiedene Kanten mit gleichem Anfangsknoten haben ungleiche Anfangsbuchstaben
- Die Beschriftung eines Weges von der Wurzel bis zum Blatt B_i ergibt $T[i \ldots n]$
- Wenn das Suffix $S[n..n]$ ein Präfix eines anderen Suffix $S[i..n]$ mit $i \neq n$ ist, dann hänge ein \$ ans Wort (ein Symbol, welches sonst nicht benutzt wird)

$\mathcal{NP}$-Vollständigkeit

$\mathcal{P}, \mathcal{NP}, \mathcal{NPH} \ldots$
$\mathcal{P} = \{\mathcal{L} | \exists$ Algorithmus A, der in polynomieller (in der Eingabegröße) Laufzeit, bestimmt ob ein $w \in L\}$ ist, bzw. eine DTM T bestimmt in polynomieller Laufzeit (Eingabegröße) ob $w \in$ L.
$\mathcal{NP} = \{\mathcal{L}|$ es existiert ein polynomieller Algorithmus A, der für alle $w \in \mathcal{L}$ für die ein x existiert mit $|x| < |w|^k$ $(k \in \mathbb{N}_0)$ bzw. $|x| \leq p(|w|)$ für $p \in \mathbb{N}[x]$, sodass $A(x, w) = 1\}$, bzw. eine NTM T die das Problem in polynomiell beschränkter Zeit löst.
$co - \mathcal{NP} = \{\mathcal{L} \,|\, \mathcal{L}^c \in \mathcal{NP}\}$
$= \{w \,|\, \forall x: |x| \leq |w|^k:\ A(x, w) = 0\}$ mit poly. Algo. A und $k \in \mathbb{N}$.
Polynomielle Reduktion Für zwei $\mathcal{L}_1, \mathcal{L}_2 \subset \Sigma^*$: $\mathcal{L}_1 \leq_P \mathcal{L}_2 \Leftrightarrow \exists f \in Abb(\Sigma^*, \Sigma^*)$, welches in polynomieller Zeit berechenbar ist, sodass $w \in \mathcal{L}_1 \Leftrightarrow f(w) \in \mathcal{L}_2$.
$\mathcal{L}$ ist $\mathcal{NP}$-schwer ($\mathcal{NPH}$), wenn $\forall \mathcal{L}' \in \mathcal{NP}: \mathcal{L}' \leq_P \mathcal{L}$, sie ist $\mathcal{NP}$-vollständig ($\mathcal{NPC}$), wenn zusätzlich noch $\mathcal{L} \in \mathcal{NP}$ ist.

$\mathcal{NP}$-Beweis
Um zu zeigen, dass $\mathcal{L} \in \mathcal{NP}$

- Algorithmus A polynomiell in $|x| + |w|$.
- Gib k (bzw. p) an
- Zwei Richtungen:
  '$\subseteq$': Für $w \in \mathcal{L}$ bel., nenne Zeugen x mit $|x| \leq |w|^k$ bzw. $\leq p(|w|)$, so daß $A(x, w) = 1$.
- '$\supseteq$': (a) Kontraposition: $w \notin \mathcal{L} \Rightarrow A(x, w) = 0\ \forall x$ oder (b) $A(x, w) = 1$, dann $w \in \mathcal{L}$.

$\mathcal{NPC}$-Probleme

SAT

Geg.: $\phi(x_1, \ldots, x_n)$ Boolsche Formel.
Ges.: Ex. erfüllende Belegung?

PARTITION

Geg.: $a_1, \ldots, a_n \in \mathbb{Z}$, $n \in \mathbb{N}$.
Ges.: $\exists I \subset \{1, \ldots, n\}$ mit $\sum_{i \in I} a_i = \sum_{j \notin I} a_j$.

SUBSET-SUM

Geg.: $S = a_1, \ldots, a_n, b \in \mathbb{Z}$, $n \in \mathbb{N}$.
Ges.: $I \subset \{1, \ldots, n\}$ mit $\sum_{i \in I} a_i = b$.

CLIQUE

Geg.: Graph G und $k \in \mathbb{N}$.
Ges.: Enthält G eine k-Clique?

INDEPENDENT-SET

Geg.: Graph G und $k \in \mathbb{N}$.
Ges.: $\exists V \in \binom{V(G)}{k}$, sodass $\forall v_1, v_2 \in V$: $\{v_1, v_2\} \notin E(G)$.

VERTEX-COVER

Geg.: Graph G und $k \in \mathbb{N}$.
Ges.: Findet man ein $I \subseteq V(G)$ mit $|I| = k$, sodass $\forall e \in E(G) : I \cap e \neq \emptyset$.

HC/HP

Geg.: (un-)gerichteter Graph G
Ges.: Ex. ein Hamiltonpfad/kreis in G?

ZOLLSTOCK

Geg.: Teilstücklängen $l_1, \ldots, l_n$ und $l \in \mathbb{N}$.
Ges.: Lässt sich der Zollstock auf eine Länge $\leq l$ falten?

SCHEDULING

Geg.: $t_1, \ldots, t_n, t \in \mathbb{N}$.
Ges.: $I \subseteq \{1, \ldots, n\}$: $\sum_{i \in I} t_i \leq t$ und $\sum_{i \in I \setminus [n]} t_i \leq t$.

KASTENPACKEN

Geg.: $s_1, \ldots, s_n \in (0, 1)$, $k \in \mathbb{N}$.
Ges.: Kann man die s_i in k Behälter mit Volumen 1 unterbringen?

TGI

Geg.: Graph G_1, G_2
Ges.: $\exists G'_2 \subseteq G_2$, sodass G'_2 isomorph zu G_1?

TSP

Geg.: $G = (V, E)$ vollständig, gerichteter, $c : E \to \mathbb{N}_0$.
Ges.: Finde Kreis, der alle Knoten genau einmal abgedeckt, mit minimalen Kosten.

Δ-TSP:
Wie TSP nur Δ-Ungleichung erfüllt und symmetrische Abstandsmatrix $D = (d_{i,j})_{1 \leq i,j \leq n}$ gegeben.

Jenseits von $\mathcal{NP}$

$t, s : \mathbb{N} \to \mathbb{N}$:
$TIME(t(n))$: Alle Sprachen, die in $t(n)$ Zeit entscheidbar sind.
$SPACE(s(n))$: In $s(n)$ Platz.
$PSPACE = \bigcup_{k \in \mathbb{N}} SPACE(\mathcal{O}(n^k))$.
$EXPTIME = \bigcup_{k \in \mathbb{N}, c \in \mathbb{R}^+} \mathcal{O}(c^{n^k})$.
$TIME(t(n)) \subset SPACE(t(n))$.
QBF ist $PSPACE$-complete. Man hat eine Formel mit Quantorausdrücken am Anfang und möchte wissen, ob dieser Ausdruck 'wahr' oder 'falsch' ist. GEO ist dies ebenfalls.

Hierarchiesätze

$s_1 = o(s_2) \Rightarrow SPACE(s_1) \subset SPACE(s_2)$.
$T_1 \log T_1 \in o(t_2) \Rightarrow TIME(t_1) \subset TIME(t_2)$.

Approximationsalgorithmen

Sei Π ein Optimierungsproblem, sei I_{min} die Optimale Lösung (für Minimierungsproblem), dann heisst ein Algorithmus A α-approximativ, wenn für alle Lösungen I_A gilt: $c(I_A) \leq \alpha c(I_{min})$.

Für kein $\alpha > 1$ existiert bei $\mathcal{P} \neq \mathcal{NP}$ ein α-approximativer Algorithmus.

Baumheuristik

Löst Δ-TSP

- Berechne MST T für G bzgl. D
- Verdoppel jede Kante von T (Eulersche Multigraph M als Ergebnis)
- Bestimme Eulerschen Weg M und konstruiere daraus Rundreise τ in G gemäß: **Lemma**: M Eulerscher Multigraph auf $V = \{1, \ldots, n\}$ mit Kosten $c(M) = \sum_{\{i,j\} \in E(M)} d_{i,j}$. Dann findet man Rundreise τ auf V in $\mathcal{O}(|E|)$ Zeit mit $c(\tau) \leq c(M)$. D.h. man iteriert den Weg und entfernt die Knoten, wenn sie nicht zum ersten Mal auftreten.

Die Baumheuristik ist 2-approximativ.

Christofides-Heuristik Algorithmus

Eingabe: G durch Abstandsmatrix $D = (d_{i,j})$ mit Δ-Ungleichung.

- Berechne MST T von G bzgl. D
- Sei $U = \{v \in V(T) \mid deg(v) \in 2\mathbb{Z} + 1\}$. Bestimme das Kostengünstigste Matching und füge Kanten hinzu (damit nun alle Knoten in T geraden Grad haben)
- Bestimme nun wieder den Eulerweg in T und bilde die Tour wie gehabt.

(F)PTAS

Ein (Fully) Polynomial-Time Approximation Scheme ist ein Algorithmus für ein Problem, der für ein $\epsilon > 0$ $1 \pm \epsilon$-approximierbar ist und für festes ϵ polynomiell in der Eingabelänge n ist. Er ist ein FPTAS, wenn er sogar polynomiell in $\frac{1}{\epsilon}$ und n ist.

PTAS für SUBSET-SUM:

- Sortiere die a_i aufsteigend.
- Bilde für $i = 1..n$ alle möglichen Summen, sortiere diese aufsteigend $\to L_i$
- Setze $\delta = \frac{\epsilon}{\delta}$ und trimme die Liste derartig, dass ein Element entfernt wird, wenn es multipliziert mit $1 - \delta$ kleiner gleich eines Vorgängers wird. Entferne alle Summen $> k$

Formeln

$H_n = \sum_{i=1}^{n} \frac{1}{n} \in \Theta(\log(n))$
$Fib(n) \geq \Phi^{n-1} = \left(\frac{\sqrt{5}+1}{2}\right)^{n-1}$
$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{\theta}{12n}}$ $\theta \in (0, 1)$
$\sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6}$
$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Anderes

Hohe W-keit:
Laufzeit effizient Las-Vegas
Ergebnis korrekt Monte-Carlo
$\log^*(n > 1) = \log^*(\log(n)) + 1$
$\log^*(n \leq 1) = 0$
$\Theta(n \log(n))$ Obj. aus $|A| = n$ wählen $\forall a \in A$ mind. 1 mal.
$T(n) \leq \sum_{i \geq 1} T(\lfloor \alpha_i n \rfloor) + \mathcal{O}(n) \in \mathcal{O}(n)$, wenn $\sum_{i \geq 1} \alpha_i < 1$ und $\alpha_i \geq 0$
$P(|X - E[X]| \geq a) \leq \frac{Var[X]}{a^2}$
$Bin(2n, n) \frac{1}{n+1} \approx 4^n$ mögliche Suchbäume mit n Knoten

Ackermannfunktion

Function $A(x, y)$
$A(0, 0) = 0, A(i, 0) = 1$
$A(0, x) = 2x$
$A(i + 1, x) = A(i, A(i + 1, x - 1))$