Technische Universität Berlin **SoSe 2015**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Dorothea Kiefer
Prof. Dr. Martin Skutella
Sören Berg, Georg Loho, Dr. Frank Lutz

Modulklausur zur Computerorientierten Mathematik I+II

20.07.2015, Beginn: 13:15 Uhr, Bearbeitungszeit: 150 Minuten

Nachname, Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Es sind keine Hilfsmittel erlaubt. Bitte nicht mit Bleistift oder mit roter Farbe schreiben!

Die Klausur ist mit mindestens ***36*** *(ITM/NidI:* ***28****)* ***Punkten*** *bestanden.*

1	2	3	4	5	6	7	8	9	10	11	Summe
8	6	12	9	5	6	8	8	6	4	8	80

1. Aufgabe (4 + 4 Punkte)

Die Komplexität der Algorithmen von Prim und Kruskal zur Bestimmung eines minimal aufspannenden Baums in einem zusammenhängenden Graphen mit nicht-negativen Kantengewichten hängt wesentlich von der jeweils verwendeten Datenstruktur ab.

- Geben Sie für beide Algorithmen eine Datenstruktur an, deren Verwendung im jeweiligen Algorithmus zu einer besonders guten Laufzeit führt.
- Beschreiben Sie jeweils die Datenstruktur anhand eines Diagramms, durch welches die Zeiger veranschaulicht werden.

Algorithmus von Prim

Datenstruktur: Heap

Erläuterung:

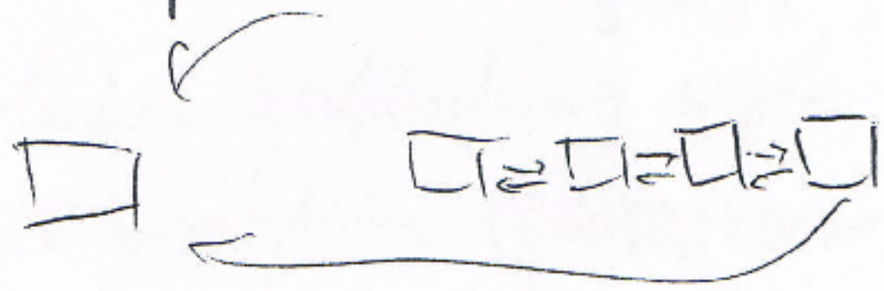
Baum mit Heap-Eigenschaft

Algorithmus von Kruskal

Datenstruktur: Union-Find

Erläuterung:

Verkettete Liste mit Zeiger auf Repräsentanten der Komponente



2. Aufgabe

(2 + 4 Punkte)

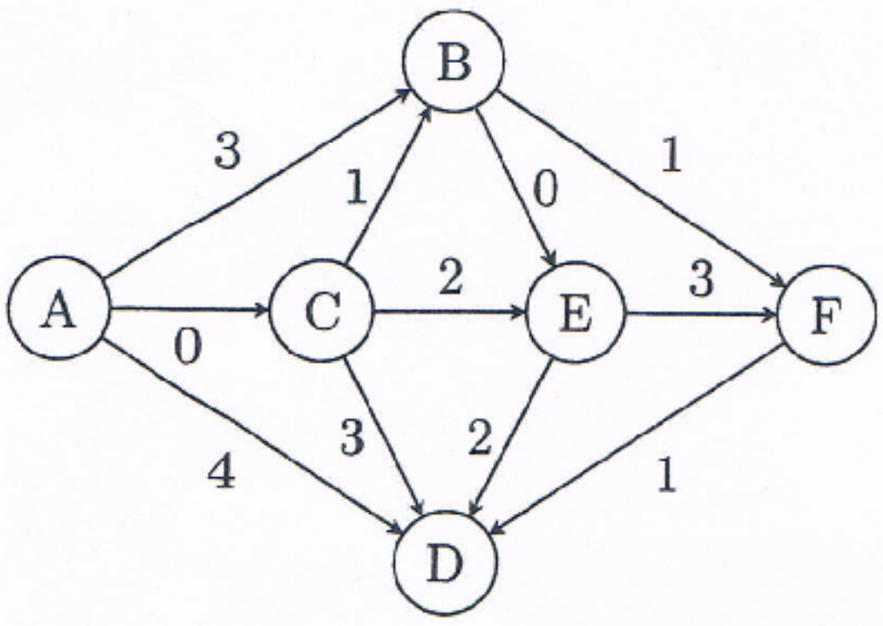
Sei $G = (V, E)$ ein gerichteter Graph mit Kantengewichten $w_e \in \mathbb{N}$, $e \in E$.

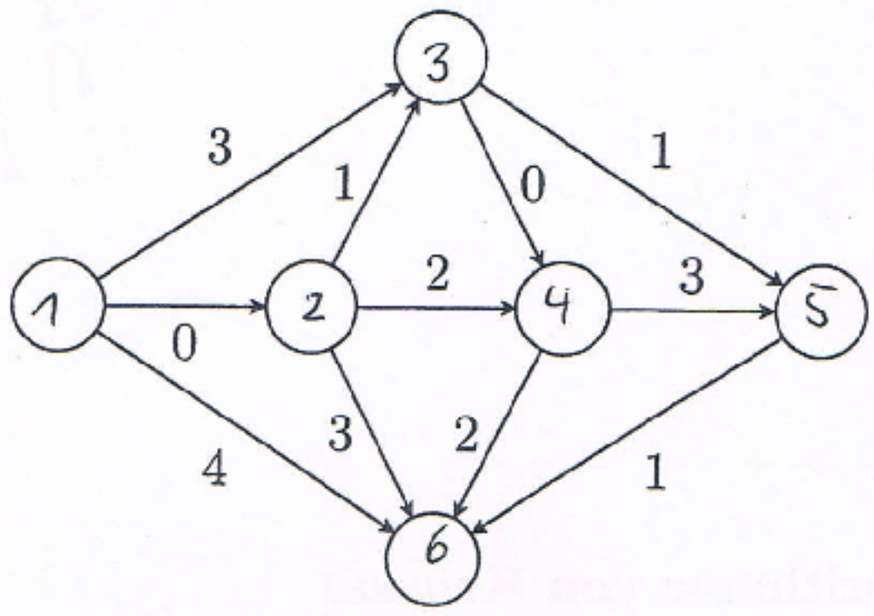
(a) Nennen Sie zwei Algorithmen zur Berechnung eines kürzesten Weges zwischen zwei gegebenen Knoten $u, v \in V$ in G und geben Sie jeweils eine möglichst kleine Komplexitätsklasse für die Worst-Case-Laufzeit (in $\mathcal{O}$-Notation) in Abhängigkeit von $n = |V|$ und $m = |E|$ an.

Ford-Fulkerson, Bellman-Ford $O(|V| \cdot |E|)$

Dijkstra $O(|E| + |V| \log |V|)$

(b) Geben Sie für den folgenden Graphen alle kürzesten Wege von Knoten A zu allen anderen Knoten an. Schreiben Sie in die Knoten auf der rechten Seite die Reihenfolge, in der die Knoten in Dijkstras Algorithmus aus der Prioritätswarteschlange extrahiert werden.





A → B

A → C

A → C → B

A → C → B → E

A → C → B → F

A → C → D

3. Aufgabe (2 + 3 + 2 + 2 + 3 Punkte)

(a) Geben Sie zwei vergleichsbasierte Sortierverfahren (außer Insertionsort und Mergesort) und deren exakte Worst-Case-Laufzeit in Θ-Notation für eine n-elementige Menge an:

Quicksort $\Theta(n^2)$ [$\Theta(n \log n)$?]

Heapsort $\Theta(n \log_2(n))$

(b) Erläutern Sie (kurz), wie man die untere Laufzeitschranke für vergleichsbasiertes Sortieren beweist.

- $n!$ Permutationen
- Binärer Entscheidungsbaum
- Höhe $\log_2(n!)$
- ~~Höhe~~ Länge bis zu Entscheidung ist Weglänge von ~~bis zur~~ Wurzel bis Blatt, also $\in \Omega(n \log_2(2))$

(c) Wie ist Sortieren in Linearzeit möglich? Nennen Sie solch einen Sortieralgorithmus.

~~Bei~~ Bucketsort : ~~Einteilung~~ Voreinteilung der zu sortierenden Menge in konstant viele Klassen

(d) Was bedeuten „stabil“ und „in-place“ bei Sortieralgorithmen?

stabil: Elemente mit gleichem Schlüsselwert werden nicht vertauscht

in-place: kein ~~zusätzlich~~ zusätzlicher Speicher benötigt

(e) Führen Sie Insertionsort für die folgende Liste durch: 1, 3, 13, 4, 6, 2.
Geben Sie hierbei die Zwischenergebnisse an.
Erläutern Sie anschließend kurz die Ideen des Algorithmus.

1 3 13 4 6 2

1 3 4 13 6 2

1 3 4 6 13 2

1 2 3 4 6 13

Sukzessives Einfügen unsortierter Elemente in den schon sortierten ersten Teil der Liste mittels binärer Suche

4. Aufgabe (4 + 1 + 2 + 2 Punkte)

(a) Geben Sie zwei Varianten von nicht-statischen Suchbäumen an. Erläutern Sie ihren Verwendungszweck und geben Sie die Laufzeiten der Operationen `insert`, `find` und `delete` an.

AVL

Splay

(b) Inwiefern ist ein optimaler statischer Suchbaum „optimal“?

Minimierung der Zugriffszeiten bei vorgegebenen Zugriffshäufigkeiten

(c) Was besagt das „Prinzip der optimalen Substruktur“ für einen statischen Suchbaum T mit Schlüsselmenge $\{s_1, \ldots, s_n\}$ und zugehörigen Zugriffshäufigkeiten $\beta_1, \ldots, \beta_n$?

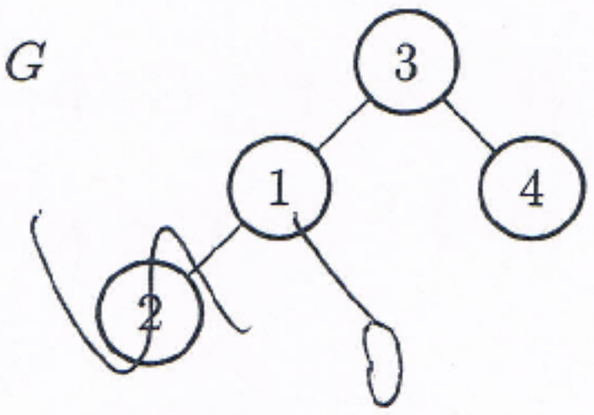
Die Summe über die Zugriffshäufigkeiten für ~~da~~ jeden ~~Tel~~ Teilbaum ist minimal bezüglich der vorkommenden Schlüsselmenge in den Teilbäumen

(d) Der folgende Graph G ist ein optimaler statischer Suchbaum für die folgende Suchschlüssel-Menge $S = \{s_1, \ldots, s_4\}$ mit Zugriffshäufigkeiten $\beta_1, \ldots, \beta_4$, wobei

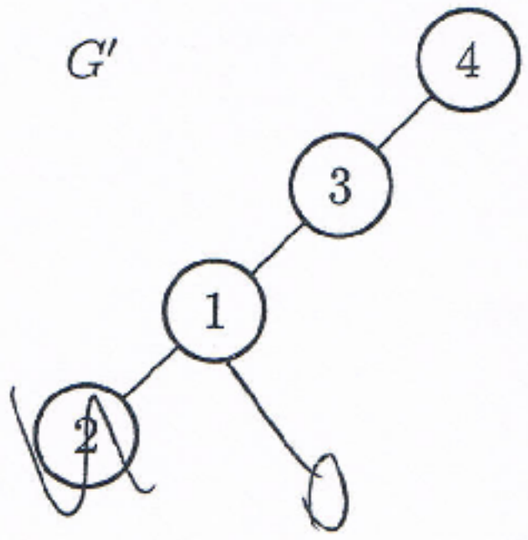
Schlüssel s_i	1	2	3	4
Häufigkeit β_i	15	10	25	50

Ist dann auch der Graph G' ein optimaler statischer Suchbaum? Begründen Sie Ihre Antwort.

G



G'

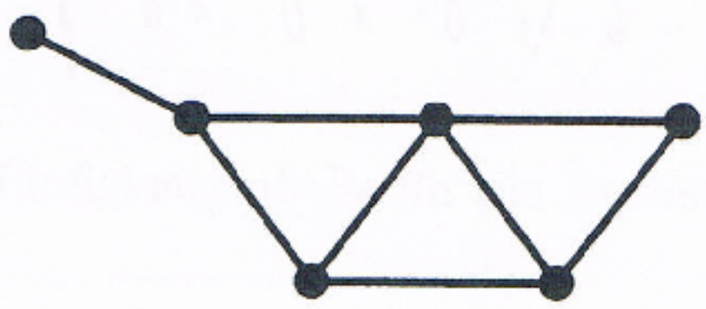


5. Aufgabe (1 + 2 + 1 + 1 Punkte)

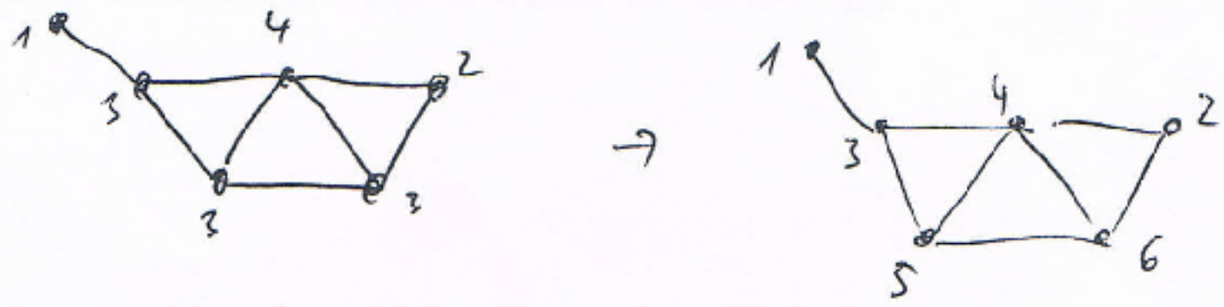
(a) Geben Sie die Definition eines „Graphenisomorphismus“ an.

bijektive Abbildung zwischen den Knotenmengen zweier Graphen, die ~~ist~~ (adjazente Knoten auf adjazente Knoten abb.) Adjazenzen erhält.

(b) Sei G der folgende (ungelabelte) Graph:



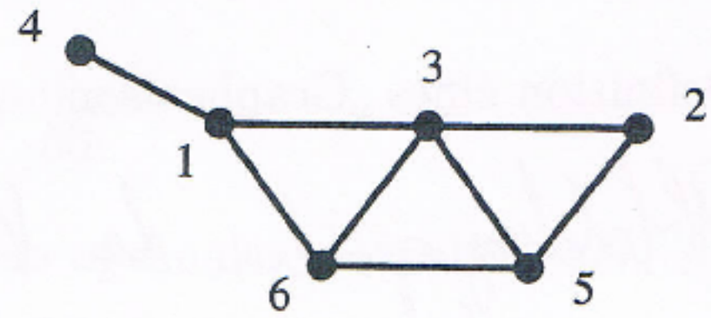
(i) Starten Sie mit der uniformen Färbung auf dem Graphen G und wenden Sie `ColorRefinement` (mehrfach) auf G an, bis eine stabile Färbung erreicht wird. Geben Sie jeweils die Färbung des Graphen an, die nach jedem Durchlauf erzielt wird.



(ii) Was können Sie mit dem Ergebnis aus (i) über die Automorphismengruppe $\mathrm{Aut}(G)$ aussagen?

besteht nur aus der Identität

(iii) Geben Sie die Adjazenzmatrix von G bei folgendem Labeling der Knoten an.



	1	2	3	4	5	6
1	0	0	1	1	0	1
2	0	0	1	0	1	0
3	1	1	0	0	1	1
4	1	0	0	0	0	0
5	0	1	1	0	0	1
6	1	0	1	0	1	0

6. Aufgabe (1 + 1 + 4 Punkte)

(a) Was ist ein Präfixcode?

(b) Wann ist ein Präfixcode optimal?

(c) Gegeben sei die folgende Häufigkeitstabelle für die Buchstaben a, b, c, d, e, f:

a	b	c	d	e	f
8	6	5	3	2	2

Wenden Sie den Huffman-Algorithmus an und geben Sie den resultierenden optimalen Präfixcode und den zugehörigen binären Baum an.

Reposition

7. Aufgabe

(2 + 3 + 3 Punkte)

(a) Nennen Sie kurz 2 Nachteile offener Adressierung im Vergleich zu anderen Hashing-Verfahren.

(b) Es sei eine leere Hash-Tabelle der Größe 7 gegeben, in der ganze Zahlen abgelegt werden sollen.

Geben Sie die Hash-Tabelle an, nachdem die Werte 10, 13, 11, 3, 8, 5, 7 in der angegebenen Reihenfolge eingefügt wurden, wobei die Hash-Funktion $h(x) = x \mod 7$ und lineares Sondieren angewendet werden soll.

(c) Es sei h eine Hash-Funktion mit gegebenem Adressraum $\{0, 1, \ldots, m-1\}$. Wir nehmen an, dass für h uniformes Hashing vorliegt. Wie hoch ist die Wahrscheinlichkeit, dass für drei unabhängig zufällig ausgewählte Elemente a, b und c aus der Menge der Schlüsselwerte die gleiche Adresse gewählt wird, d. h. dass $h(a) = h(b) = h(c)$ gilt?

8. Aufgabe

(4 + 2 + 2 Punkte)

Gegeben sei die folgende deterministische Turingmaschine mit Startzustand q_0, Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4\}$, Eingabealphabet $\Sigma = \{0, 1\}$, Bandalphabet $\Gamma = \{0, 1, x, y, B\}$ und Menge akzeptierender Endzustände $F = \{q_4\}$:

δ	0	1	x	y	B
q_0	$(q_1, x, \mathtt{R})$	–	–	$(q_3, y, \mathtt{R})$	–
q_1	$(q_1, 0, \mathtt{R})$	$(q_2, y, \mathtt{L})$	–	$(q_1, y, \mathtt{R})$	–
q_2	$(q_2, 0, \mathtt{L})$	–	$(q_0, x, \mathtt{R})$	$(q_2, y, \mathtt{L})$	–
q_3	–	–	–	$(q_3, y, \mathtt{R})$	$(q_4, B, \mathtt{R})$
q_4	–	–	–	–	–

Hier bedeutet '–' in Zeile q und Spalte a den Eintrag $(q, a, \mathtt{N})$.

(a) Vervollständigen Sie jeweils die folgenden beiden Konfigurationsfolgen. Geben Sie auch an, ob die deterministische Turingmaschne für die jeweilige Konfigurationsfolge in einem akzeptierenden Endzustand terminiert.

(i) $Bq_00011 \vdash xq_1011 \vdash x0q_111 \vdash xq_20y1 \vdash Bq_2x0y1$
$\vdash xq_00y1 \vdash xxq_1y1 \vdash xxyq_11 \vdash xxq_2yy \vdash xxyq_2y$
$\vdash xxq_0yy \vdash xxyq_3y \vdash xxyyq_3B \vdash xxyyBq_4B$ $\vdash xxyq_2yy$

akzeptiert

(ii) $Bq_00010 \vdash xq_1010 \vdash x0q_110 \vdash xq_20y0 \vdash Bq_2x0y0$
$\vdash xq_00y0 \vdash xxq_1y0 \vdash xxyq_10 \vdash xxy0q_1B$

nicht akzeptiert

(b) Welche Sprache $L \subseteq \Sigma^*$ entscheidet die obige deterministische Turingmaschine? Es genügt, die gesuchte Sprache ohne Beweis anzugeben.

$L = \{0^n 1^n : n \geq 1\}$ Achtung: Leeres Wort wird nicht akzeptiert, i.e. 0^n1^n für $n=0$.

(c) Ändert sich die durch die obige deterministische Turingmaschine entschiedene Sprache, falls auch q_3 ein akzeptierender Endzustand ist, d. h. falls gilt $F = \{q_3, q_4\}$? Falls nein, begründen Sie kurz, warum sich die Sprache nicht ändert. Falls ja, geben Sie ein Wort an, für welches die Turingmaschine nun in einem akzeptierenden Endzustand hält, zuvor jedoch in einem nicht akzeptierenden Zustand terminierte.

Ja, z.B. 010, 011, ...

?

9. Aufgabe (6 Punkte)

Kreuzen Sie jeweils an, ob die folgenden Aussagen wahr oder falsch sind. Sie müssen Ihre Wahl nicht begründen.

Falsch angekreuzte Teilaufgaben werden abgezogen, wobei die Aufgabe insgesamt mit mindestens 0 Punkten bewertet wird.

(a) Sind f und g zwei Funktionen, so gilt $f = O(g)$ oder $g = O(f)$.

☐ Wahr ☒ Falsch Gegenbeispiel $f = \sin$, $g = \cos$

(b) Der Algorithmus von Bellman–Ford kann in Laufzeit $O(n^3)$ implementiert werden, wobei n die Knotenanzahl des Graphen bezeichne.

☒ Wahr ☐ Falsch klar

(c) Ist $G = (V, E)$ ein azyklischer Digraph, so kann das Kürzeste-Pfade-Problem für G in Laufzeit $O(|V| + |E|)$ gelöst werden.

☒ Wahr ☐ Falsch → topol. Sortierung

(d) Ist A ein Array bestehend aus n ganzen Zahlen, wobei jede Zahl im Intervall $(-e^{10}, 10^6]$ liegt, so kann A in Linearzeit $O(n)$ sortiert werden.

☒ Wahr ☐ Falsch Radixsort

(e) Bei universellem Hashing ändert sich die Hash-Funktion jedesmal, wenn ein Element hinzugefügt wird.

☐ Wahr ? ☒ Falsch Hashfkt. wird vor Anwendung zufällig gewählt. Danach 'fest'.

(f) Das Halteproblem ist nicht rekursiv.

☒ Wahr ☐ Falsch (Theorem aus der VL)

10. Aufgabe

(2 + 2 Punkte)

Es seien k sortierte Listen ganzer Zahlen gegeben, sodass die Anzahl der Elemente in allen Listen zusammen n ist.

Durch wiederholte Anwendung der Operation `merge` für jeweils zwei Listen kann man die Listen in einer gesamten, sortierten Liste vereinen. Die Laufzeit für dieses Verfahren beträgt $\Theta(nk)$.

Es soll nun eine Möglichkeit beschrieben werden, wie man die k Listen mit Hilfe eines Heaps in Laufzeit $\Theta(n \log k)$ zu einer gesamten, sortierten Liste zusammenfügen kann. Der gesuchte Algorithmus verwendet einen binären Min-Heap und benötigt damit nur $\Theta(k)$ zusätzlichen Speicherplatz.

(a) Welche Zahlen sollten im Heap gespeichert werden? Bedenken Sie, dass nur $\Theta(k)$ zusätzlicher Speicherplatz verbraucht werden soll.

die Minima der Listen

(b) Welche Zahl sollte nach der Durchführung der Operation `ExtractMin` während des Algorithmus als nächstes im Heap auftauchen?

die nächst kleinere Zahl der Liste, deren Minimum gerade gelöscht wurde

11. Aufgabe

(4 + 4 Punkte)

Gegeben sei ein Array A mit $n \geq 3$ paarweise verschiedenen Zahlen und es sei $D = \{\, |v - w| : v, w \in A \,\}$ die Menge der paarweisen Abstände der Zahlen aus A.

Schreiben Sie für die folgenden zwei Probleme je einen Algorithmus in Pseudocode oder Python, wobei Algorithmen aus der Vorlesung als gegebene Funktionen vorausgesetzt werden dürfen.

(a) Gesucht ist ein Algorithmus mit Worst-Case-Laufzeit $O(n)$: Es sollen *Indizes* für zwei Elemente $x, y \in A$ bestimmt werden, sodass $|x - y|$ das Maximum von D ist.

```
INPUT : Array A
OUTPUT: Paar von Indizes
n = 0 , 0
for i = 0 to len(A):
    for j = 0 to len(A):
for j in range(len(A))
        if abs(A[i] - A[j]) > n:
            n = abs(A[i] - A[j])
return m1, m2
```

(b) Gesucht ist ein Algorithmus mit Worst-Case-Laufzeit $O(n \log n)$: Es sollen zwei Elemente $x, y \in A$ gefunden werden, deren Abstand $|x - y|$ das zweit-kleinste Element in D ist.

```
INPUT : Array A
OUTPUT: ~~Paar von Zahlen aus A~~ Zahl aus A
k = ∞
z = ∞
L = sorted(A)
for x, y in zip(L[0:], L[1:]):
    M = sorted([k, z, y-x])
    k, z = M[0], M[1]
for x, y in zip(L[0:], L[2:]):
    M = sorted([k, z, y-x])
    k, z = M[0], M[1]
```