

8
(min-heap) Heaps

keys for Items

Insert (x, Q) min(Q)

Delete min(Q) DecreaseKey(x, Q, Δ)

decrease priority

insert | delete

$O(\log n)$ > ...

$O(1)$ > min

DecreaseKey

amortized $O(1)$

Dijkstra

priority queue

step

priority queue

priority queue

priority queue

priority queue

Dijkstra

$\delta(u,v)$ is the shortest path from u to v .
 We want to find $\delta(s, t)$.
 We maintain a priority queue Q of vertices v with their current distance estimate $d[v]$.
 Initially $d[s] = 0$ and $d[v] = \infty$ for all other v .

While Q is not empty:
 1. Extract u from Q .
 2. For each neighbor v of u :
 If $d[v] > d[u] + c(u,v)$, then update $d[v]$ and insert v into Q .

The algorithm terminates when t is extracted from Q .
 At this point, $d[t]$ is the shortest path distance from s to t .
 The set of vertices S that have been processed is $S = \{v \mid v \text{ has been extracted from } Q\}$.

The correctness of Dijkstra's algorithm relies on the fact that the shortest path from s to any vertex v in S consists only of vertices in S .
 This is true because the algorithm always extracts the vertex with the smallest distance estimate, and any path to a vertex not yet extracted would have a larger distance estimate.

The time complexity of Dijkstra's algorithm is $O(V^2)$ if implemented with an adjacency matrix and a simple array for the priority queue.
 It can be improved to $O(E \log V)$ using a binary heap or Fibonacci heap.

The algorithm is used to find the shortest path between two vertices in a weighted graph.
 It is a special case of the more general Bellman-Ford algorithm.

von $d[s, t]$
 ist die kürzeste Distanz
 von s nach t .
 Wir wollen $d[s, t]$ berechnen.
 Wir führen eine Prioritätswarteschlange Q mit den Knoten v und ihren aktuellen Distanzwerten $d[v]$.
 Anfangs $d[s] = 0$ und $d[v] = \infty$ für alle anderen v .

Solange Q nicht leer ist:
 1. Nimm u aus Q .
 2. Für jeden Nachbarn v von u :
 Wenn $d[v] > d[u] + c(u,v)$, dann aktualisiere $d[v]$ und füge v in Q ein.

Der Algorithmus endet, wenn t aus Q entnommen wird.
 Zu diesem Zeitpunkt ist $d[t]$ die kürzeste Distanz von s nach t .
 Die Menge der Knoten, die bereits bearbeitet wurden, ist $S = \{v \mid v \text{ wurde aus } Q \text{ entnommen}\}$.

Die Korrektheit von Dijkstras Algorithmus beruht auf der Tatsache, dass die kürzeste Distanz von s nach einem Knoten v in S nur aus Knoten in S besteht.
 Dies ist richtig, weil der Algorithmus immer den Knoten mit dem kleinsten Distanzwert entnimmt, und jede alternative Route zu einem noch nicht entnommenen Knoten einen größeren Distanzwert hätte.

for each element in the array

insert element

Insert

min

delete min

Decrease key

for each element in the array

$h \leq \log_2(n)$

insert element

Insert $h = \lfloor \log_2 n \rfloor$

min h

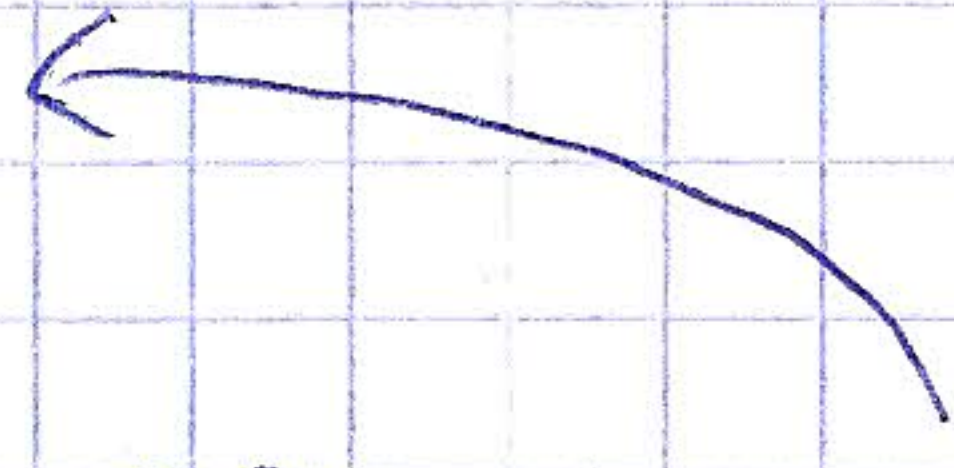
delete min h

decrease key $m = |E|$

insert element $O(\log n)$

$h \log h$

$O(m + n \log h)$

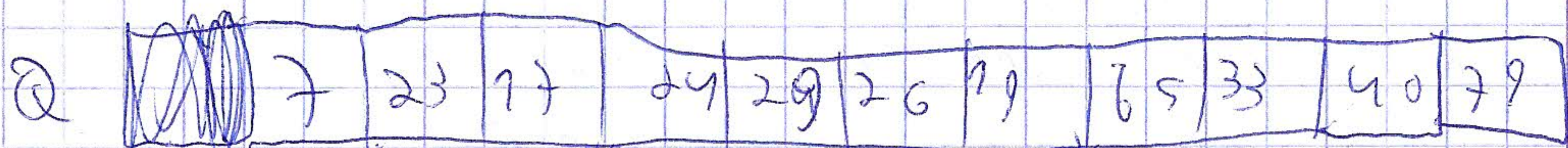


Heap order

insert element

delete min

decrease key



I have B have M have

מבנה נתון

מבנה נתון : delete min

מבנה נתון : delete min

Heapify down

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

Heap Order

Heapify down

מבנה נתון : delete min

Insert

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

$O(n)$ find min

מבנה נתון : delete min

$O(\log n)$ delete min

מבנה נתון : delete min

$O(\log n)$ insert

מבנה נתון : delete min

deleted

decrease key (x, Q, A)

מבנה נתון : delete min

↑

מבנה נתון : delete min

heapify up

Heapify

מבנה נתון : delete min

מבנה נתון : delete min

Heapsort (Williams, Floyd, 1964)

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

מבנה נתון : delete min

in-place

Heap of nodes

Heapify up



Heapify down

Heapify

inplace

Heapify

Heapify

Heapify

Heapify

Heapify

$$\frac{x}{(1-x)^2}$$

$$\frac{1}{(1-x)^2}$$

$$1+x+x^2+\dots$$

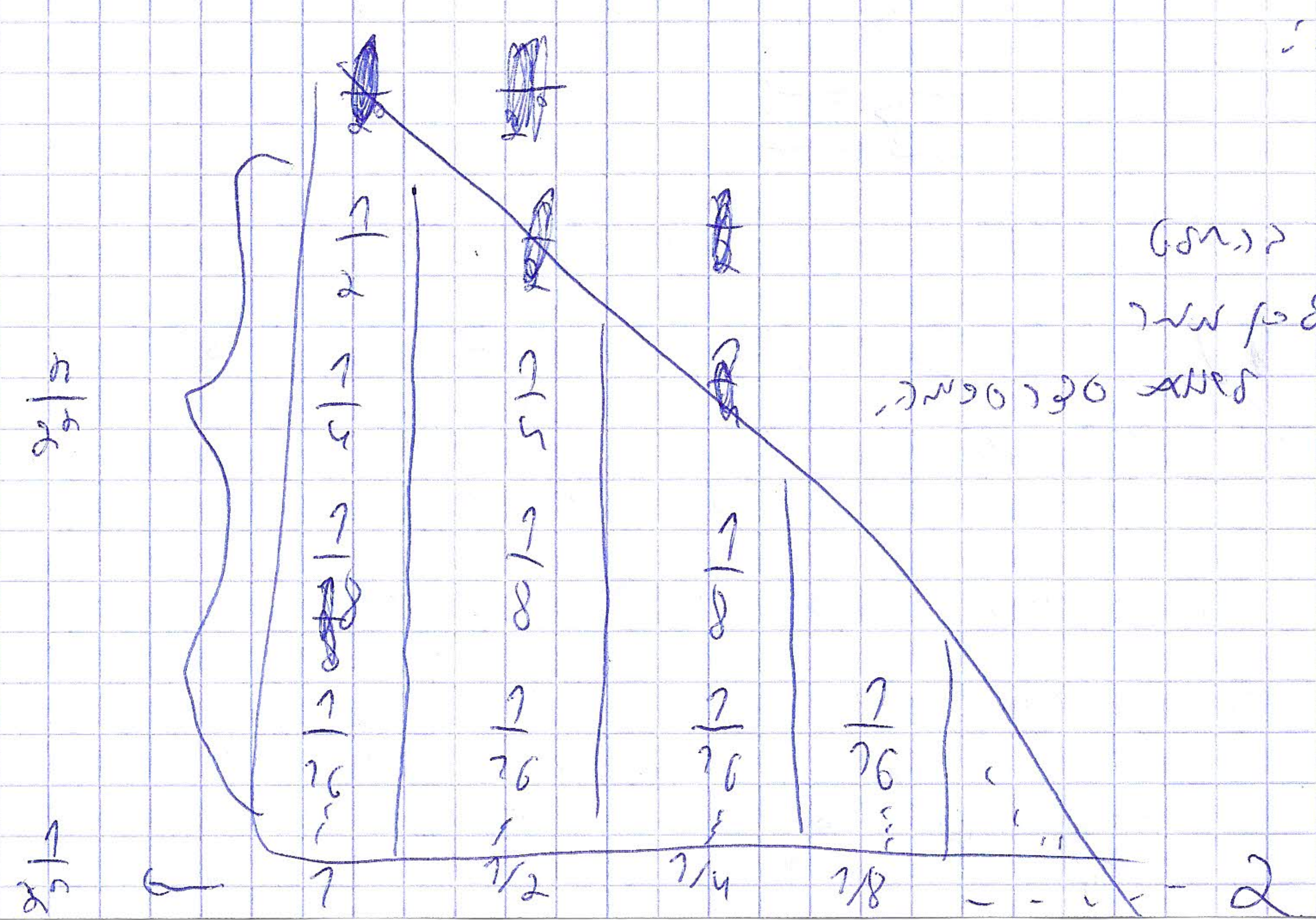
$$\sum_{k=0}^{\infty} kx^{k-1} = x \sum_{k=0}^{\infty} kx^{k-2}$$

$$\frac{x}{(1-x)^2}$$

$$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \dots \leq n \sum_{k=1}^{\infty} \frac{k}{2^k}$$

$$\sum_{k=0}^{\infty} \frac{k}{2^k} = \sum_{k=0}^{\infty} \frac{1}{2^k}$$

Heapify

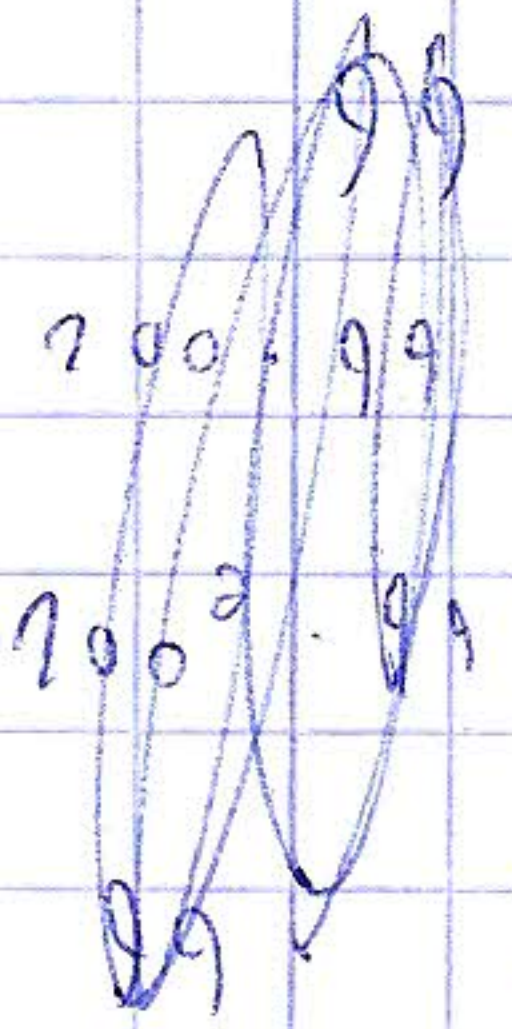


Heapify

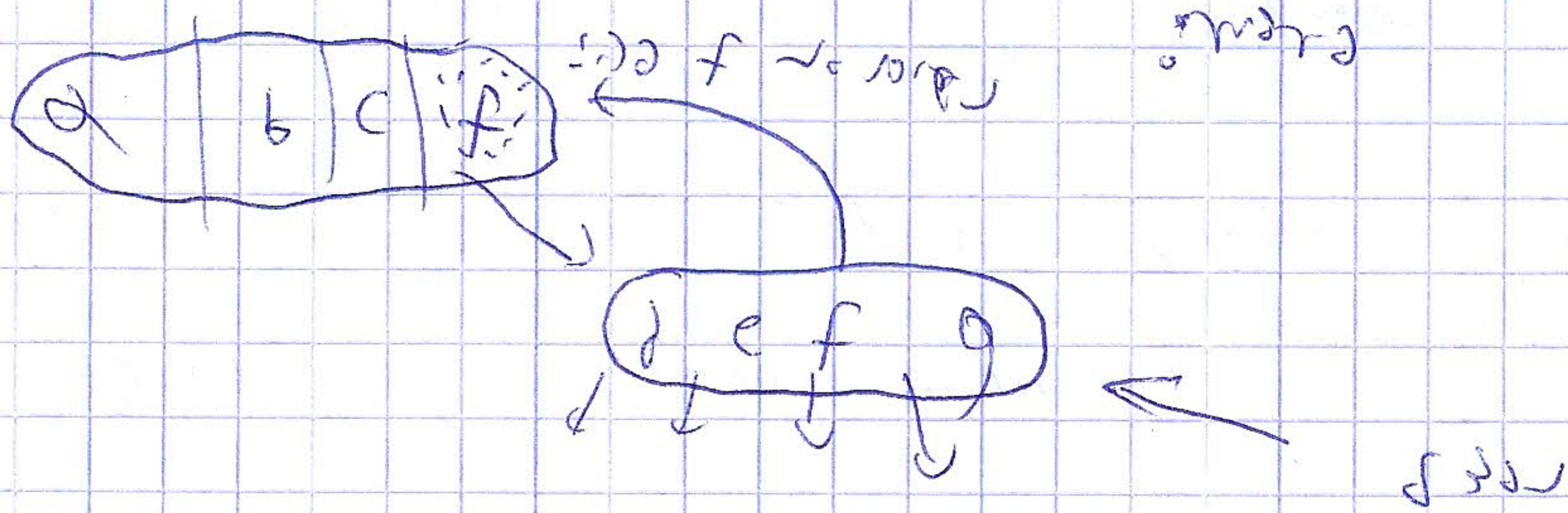
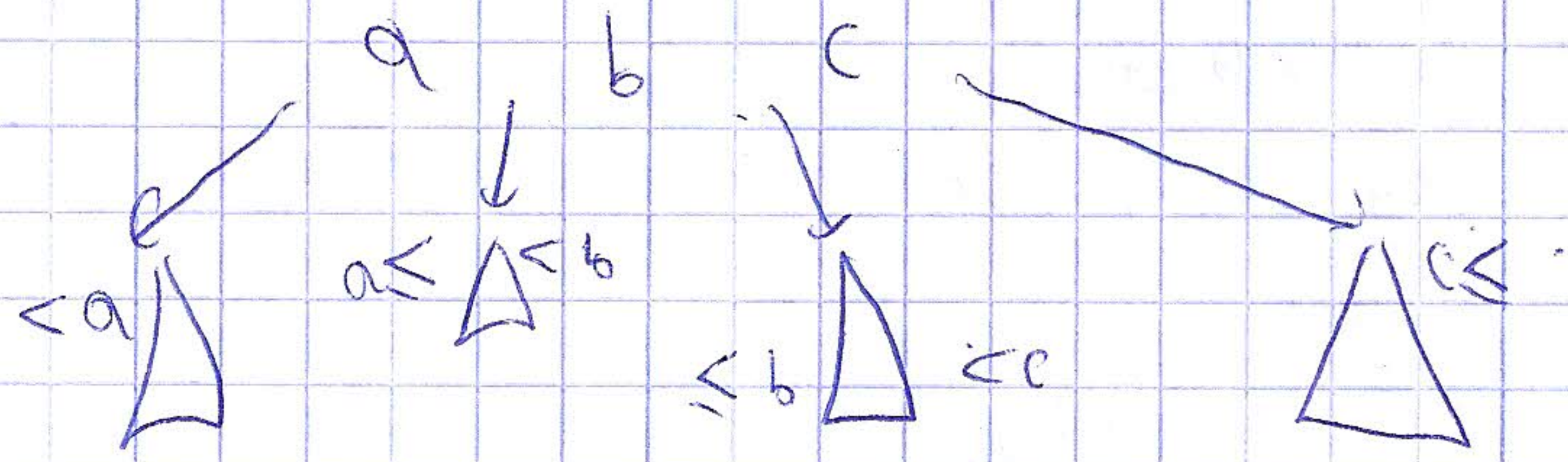
50%

~~99(1+1000)~~

B Trees



$n \geq 2k$ $n \geq 2k$ $n \geq 2k$ $n \geq 2k$ $n \geq 2k$ $n \geq 2k$ $n \geq 2k$
 $n \geq 2k-1$ $n \geq 2k-1$ $n \geq 2k-1$ $n \geq 2k-1$ $n \geq 2k-1$ $n \geq 2k-1$ $n \geq 2k-1$



rest nodes a b c f n n n n n n n n

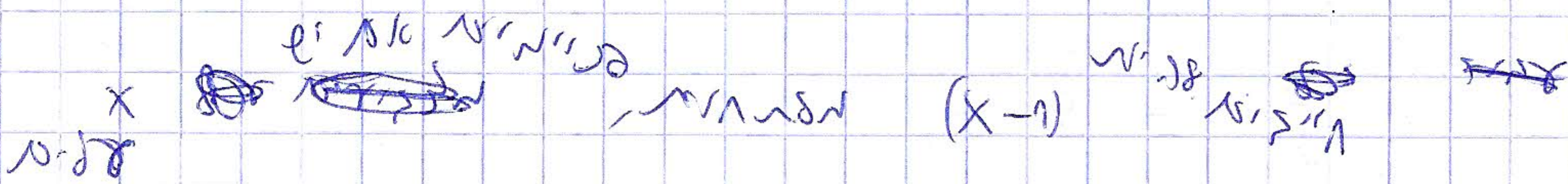
... so ... 50-1000, k=50 B Tree

... (minimum) ...

...

...

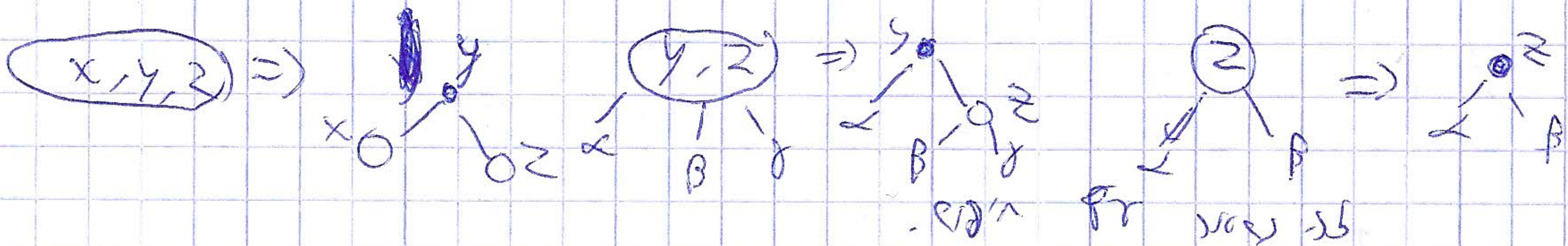
$$99x + (x-1) > 99,999$$



$$100x \geq 100,000$$

$$x \geq 1000$$

... 2-4 ...



... 2-4 ...

... B Tree



~~Handwritten scribbles at the top of the page.~~

Handwritten notes: "דבר זה הוא..."

Handwritten notes: "Tree & Graph..."

Handwritten notes: "B Tree", "insert", "split", "S_i", "S_i^a".

Handwritten notes: "Phi(D)", "split", "S_i^a".

$$\Phi(D_{i+1}) \leq \Phi(D_i) + 1$$

Handwritten notes: "split", "S_i^a".

$$S_i^a = S_i^a + \Delta \Phi(D) \leq 1$$

$$\Delta \Phi(D) \leq S_i - S_i^a$$

Handwritten notes: "S_i", "Phi", "S_i^a".