Exercise sheet 7.

Advanced Algorithms WiSe 2020/21

Instructor: László Kozma

Due 12:00, January 8th, 2020

Exercise 1 Approximate Counting *4 + 4 Points*

In the lecture we implemented the *count-min* data structure as follows:

Process(a_j, c_j)
  for $i \in \{1 \dots k\}$ **do**
    $C[i, h_i(a_j)] \leftarrow C[i, h_i(a_j)] + c_j$

Count(x)
  $c^* = \min_{i=1}^{k} C[i, h_i(x)]$
  return c^*

Consider the following variant of processing the elements:

Process1(a_j, c_j)
  `min` $\leftarrow \min_{i=1}^{k} (C[i, h_i(a_j)] + c_j)$
  for $i \in \{1 \dots k\}$ **do**
    $C[i, h_i(a_j)] \leftarrow \max\{C[i, h_i(a_j)], \texttt{min}\}$

In Process1 we do not necessarily increase *all* counters by the full amount, only those that have the smallest values. All other counters are increased so that they have at least the same value. Specifically, all counters with a value larger than `min` are not changed at all and all counters with a value smaller than `min` are set to `min`.

Let $c^*(x)$ and $c_1^*(x)$ be the results of Count(x) for the data structure built using Process and Process1 respectively. Let `count`(x) denote the actual (true) count for $x \in U$. Show that the theorem from class also holds for $c_1^*(x)$:

(a) Show that `count`$(x) \leq c_1^*(x)$

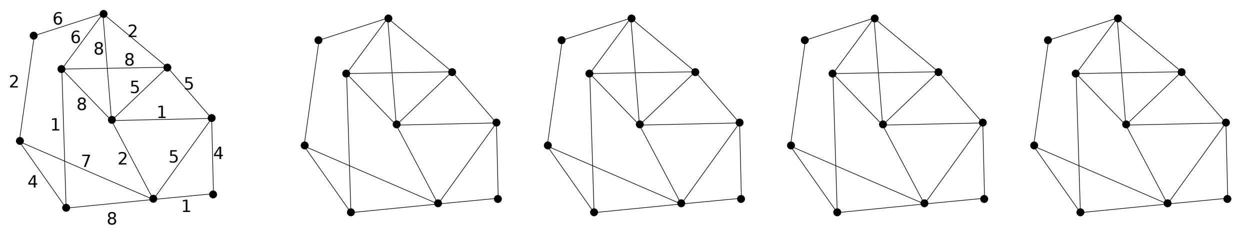
(b) Show that $c_1^*(x) \leq$ `count`$(x) + \varepsilon \sum_{t=1}^{T} c_t$ with probability at least $1 - \left(\frac{1}{\ell\varepsilon}\right)^k$, where k is the number of hash functions $h_i : U \to \{0, \dots, \ell - 1\}$.
Note that it is sufficient to show $c_1^*(x) \leq c^*(x)$.

(c) (*4 bonus points*) Implement the two variants of the count-min data structure, and run some experiments with different input streams (a_j, c_j) of your choice (either synthetic or some real-world data). Especially consider streams where there is a large discrepancy between the largest and smallest counts.

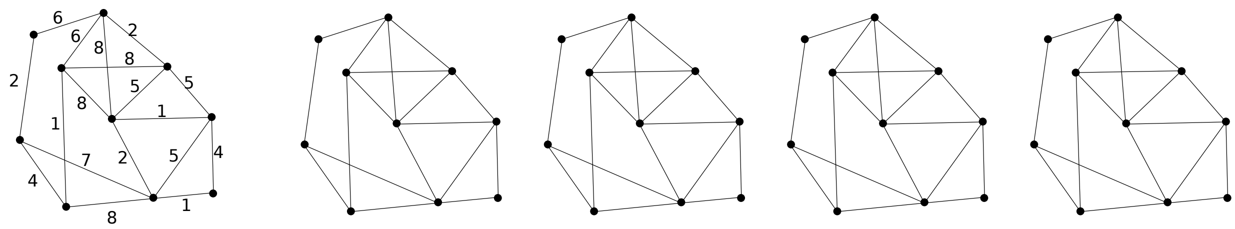
Make some observations about the quality of the approximation in both cases and report your findings. You may also think of and experiment with other variations on the data-structure.

Exercise 2 Minimum Spanning Trees *8 Points*

(a) (2 pts) Show the execution steps of Borůvka's algorithm on the following graph.



(b) (2 pts) Show the execution steps of one other algorithm of your choice.



(c) (4 pts) Suppose we have computed the MST of a graph. Now the weight of an edge that is not in the MST changes. Can you update (if necessary) the MST faster than recomputing it from scratch?
Hint: Which edge(s) may need to be removed from the MST?

Can you also update the MST efficiently if the weight of an edge in T changes?
Hint: Which edge(s) may need to be added to the MST?

(d) (optional 2 pts) Prove that if all edge-weights in a graph are distinct, then the minimum spanning tree (MST) is unique.

Total: 16 points. Have fun with the solutions and enjoy the holiday!

From: Yumeng Li and Thore Brehmer

Ex sheet 7 Ex1.

(a) Induction: $count(x) \le C_1^*(x)$.

Assume we process $(a_1, c_1), \ldots (a_T, c_T)$

If $T=1$. trivial.

Assume true for $T=n-1$. WTS true for $T=n$.

For $x \ne a_T$, $count(x) \le C_1^*(x)$ by induction.

when process (a_T, c_T), $count(x)$ remains the same but $C_1^*(x)$ could increase, therefore $count(x) \le C_1^*(x)$ for $T=n$.

If $x = a_T$, then In PROCESS1(a_T, c_T), we have for all i

$C[i, h_i(x)] \ge min$.

where $min \leftarrow min_{i=1}^{k} (C[i, h_i(x)] + c_T) = min_{i=1}^{k} (C[i, h_i(x)]) + c_T$

$= C_1^*(x) + c_T$

$C_1^*(x)$, $C[i, h_i(x)]$ are the number before processing (a_T, c_T)

$\rightsquigarrow$ by $C_1^*(x) = min_{i=1}^{k} (C[i, h_i(x)])$ we know after processing (a_n, c_n) for $n=T$, $C_1^*(x)$ increase at least c_T, and $count(x)$ increases exactly c_T. So by induction $count(x) \le C_1^*(x)$ for $T=n$

(b) It suffices to show $C_1^*(x) \le C^*(x)$.

Since $C^*(x) = min_{i=1}^{k} C[i, h_i(x)]$, $C_1^*(x) = min_{i=1}^{k} C_1[i, h_i(x)]$

where we denote the table in PROCESS1 by $C_1[i, h_i(a_j)]$.

It suffices to show $C[i, h_i(x)] \ge C_1[i, h_i(x)]$ for all i.

It's trivial if $T=1$. Assume true after we process (a_i, c_i) $1 \le i \le n-1$, when process (a_n, c_n), we have

$C[i, h_i(a_n)] \leftarrow C[i, h_i(a_n)] + c_n$

$C_1[i, h_i(a_n)] \leftarrow max\{C_1[i, h_i(a_n)], min\}$.

where $C_1[i, h_i(a_n)]$ is the value before processing (a_n, c_n)

If $C_1[i, h_i(a_n)] = C_1[i, h_i(a_n)]$ then by induction

$C_1[i, h_i(a_n)] = C_1[i, h_i(a_n)] \le C[i, h_i(a_n)] \le C[i, h_i(a_n)]$

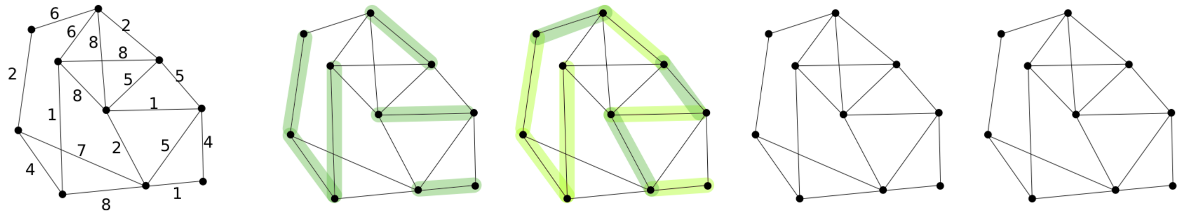
If $C_1[i, h_i(a_n)] = min = min_{i=1}^{k} (C_1[i, h_i(a_n)] + c_n)$

$\le C[i, h_i(a_n)] + c_n = C[i, h_i(a_n)]$

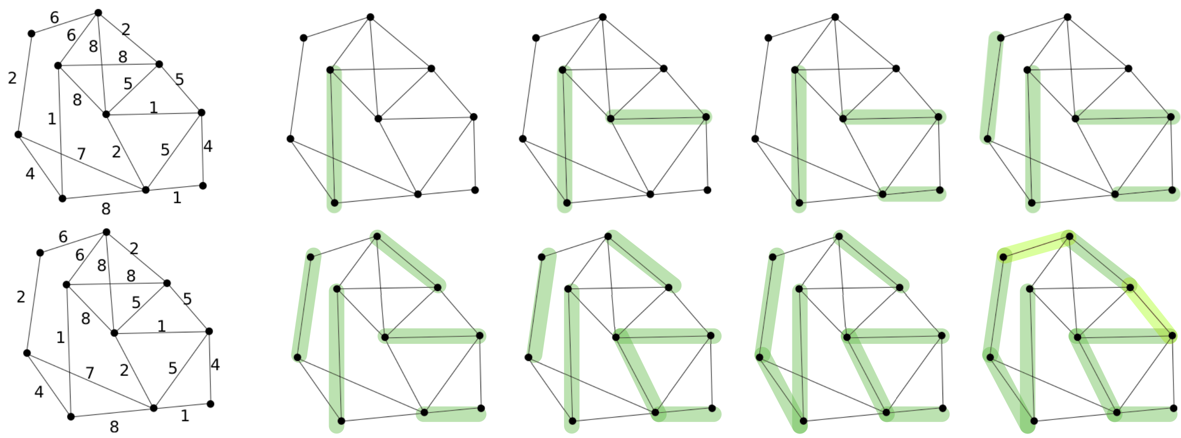
$\Rightarrow C_1^*(x) \le C^*(x)$.

Exercise 2 Minimum Spanning Trees *8 Points*

(a) (2 pts) Show the execution steps of Borůvka's algorithm on the following graph.



(b) (2 pts) Show the execution steps of one other algorithm of your choice.



(c) (4 pts) Suppose we have computed the MST of a graph. Now the weight of an edge that is not in the MST changes. Can you update (if necessary) the MST faster than recomputing it from scratch?
Hint: Which edge(s) may need to be removed from the MST?
Can you also update the MST efficiently if the weight of an edge in T changes?
Hint: Which edge(s) may need to be added to the MST?

c) 1. Edge in MST

1.1. Edge value was increased => MST is still MST

1.2. Edge value was decreased

O(1) - Add edge to the MST

O(n) - Find the circle with the added edge (e.g. with bfs or dfs)

O(n) - Remove the edge with the highest value

in worst case O(n), if the circle contains all n edges

- but is often faster than recomputing from scratch

2. Edge not in MST

2.1. Edge value was decreased ⇒ MST is still MST

2.2. Edge value was increased

$O(1)$ - remove edge from MST

- there will now be 2 components

$O(n)$ - find the edge with the lowest value

↳ in worst case, as the components could have all (n) possible edges between them

- but is often faster than recomputing from scratch

(d) (optional 2 pts) Prove that if all edge-weights in a graph are distinct, then the minimum spanning tree (MST) is unique.

Assume we have 2 distinct minimum MST's T_1 and T_2.

Let the edge $e_1 \in T_1 \wedge e_1 \notin T_2$

If we add e_1 to T_2, there will be a cycle.
So T_2 will no longer be a minimum MST.

To regain the minimum MST property we remove the edge r
with the highest value from the cycle.

Case 1. $r = e_1 \Rightarrow$ T_2 has an edge $< e_1$

$\Rightarrow$ The total weight of $T_2 < T_1$

$\Rightarrow$ T_1 can not be minimum MST

case 2. $r \neq e_1 \Rightarrow$ $e_1 < r$

$\Rightarrow$ T_1 has an edge $< r$

$\Rightarrow$ The total weight of $T_1 < T_2$

$\Rightarrow$ T_2 can not be minimum MST

$\Rightarrow$ if all edges weights are distinct, then there can
only be one minimum MST (unique)