Padovan heaps

Vladan Majerech
Main principle – superexpensive comparisons

• We compare as late as possible (FindMin does almost all work).
• We never forget results of comparisons.
• Organization of comparisons should not take asymptotically more time than comparisons.
Invariant of c-q narrow trees

There will be rank defined for a heap element $v$ denoted $\text{rank}_v$. (New element gets rank 0).

- For $c>0$ and $1<q\leq 2$ holds:

Subtree of ancestors of an element of rank $k$ has size at least $cq^k$. 
Saving one pointer per element

• Decrement in Fibonacci heaps require pointer to successor (parent) to allow rank decrement propagation.
• We save the space by replacing nil pointers at the right ends of lists by pointer to successor (parent). We would not maintain pointers to successors (parents) on other places.
• In that case we cannot propagate rank decrements except at the right end of the list where we can access the successor.
• We would propagate rank recomputation only at the rightmost two elements (elements of highest rank among white and black).
• Worst case time for DeleteMin will be O(1) again.
rank_v and wrank_v

• Let us define \( wrank = 1 + rank \) for a black element, and \( wrank = rank \) for a white one. Let \( wrank = -1 \) for \textit{null} pointers (missing vertices).

• Predecessor (children) lists have all red on left, than \( wrank \) increasing (when defined).

• Take among black and white predecessors last two ... \( w_0, w_1 \) from right (they could be missing).

  S. Safe: \( wrank_{w_1} + 1 = wrank_{w_0} \): \( rank_v = wrank_{w_0} + 1 \).

  D. Dangerous: \( wrank_{w_1} + 1 < wrank_{w_0} \) and \( w_0 \) is white: \( rank_v = wrank_{w_0} \). (\( v \) cannot be white)

  F. Forbidden: \( wrank_{w_1} + 1 < wrank_{w_0} \) and \( w_0 \) is black: color \( w_0 \) yellow and recompute.
Minimal size recurrence
Narrowsness

• Let $M_k$ be minimal size of an ancestors tree of an element of rank $k$
• Following recurrence holds:
  $$ M_k = 1 + M_{k-2} + M_{k-3} $$
• This is closely related to Padovan sequence and c-q narrowsness holds
  for $q = \frac{3}{2} \left( 1 + \sqrt{\frac{23}{27}} \right) + 3 \sqrt{\left( \frac{1}{2} \left( 1 - \sqrt{\frac{23}{27}} \right) \right)} \approx 1.324718$
• $(q^3 = q + 1)$
Decrement fully colored

• We cut an element and if it was one of last two elements of a list we call cascading rank consolidation on it’s successor (parent).

• During cascade consolidation we start by computation of current rank. If there is no change, we are done.

• If rank of a white element drops by 1, it is colored black.

• If rank of a black element drops, it is colored yellow as well as if rank of a white element drops by at least 2.

• If an element is among last two of the list we continue the rank consolidation at the successor (parent).

• During rank calculation of an element we move all yellow predecessors (children) among right two elements to the left end. We color them red during it. If there is a red element among rightmost two, we know there is no more white and black element to the left in the list.
Picture with all colors (negated keys)
Summary

- Potential used in analysis is
  \[ \Phi_0 t_0 + \Phi_1 t_1 + \Phi_2 t_2 + 2\Phi_3 t_3 + \Phi_4 t_4 + \Phi_5 t_5 + \Phi_6 t_6, \]
  where \( t_0 \leq t_1 < t_2, t_1 < t_5 < t_6, t_5 + t_6 < t_3, \) and \( t_5 + t_6 < t_4 \) are appropriate constants.

- \( \Phi_0 \) is number of candidates for minimum.
- \( \Phi_2 \) is number of candidates for minimum, bounded by highest achievable rank + 1.
- \( \Phi_1, \Phi_3, \) resp. \( \Phi_5 \) is number of all red, black, resp. yellow descendants (children).
- \( \Phi_4 \) is sum of differences between rank and number of white and black descendants (children) of an element.
- \( \Phi_6 \) is number of dangerous vertices
The payment schema
Data structure’s competition

• Supporter of one structure generates sequence of method calls, both structures invoke the methods and ratio of total times is the gain. The game is repeated with roles changed.

• Heaps according superexpensive comparison principle will won against standard implementation by $\theta(\log n)$ using prefix of a sequence

  $i=0;\text{Repeat (Insert(-i), Insert(-(i+1)), FindMin, DeleteMin, i++),}$

  because maximal rank achieved would be 4.

• Standard heaps could gain at most $\theta(1)$ in revenge.