# Algorithms Example Sheet 6: Core Questions

## Dynamic Array

**Exercise 6.C.1** Consider the dynamic array from section 7.3 in lecture notes, but suppose that when the array needs to be expanded we multiply the capacity by a factor $k > 1$ rather than doubling. What is wrong with the following argument?
*"Define the potential to be $\Phi = 2n - \ell$, where $n$ is the number of items and $\ell$ is the capacity, with the special case $\Phi = 0$ when $n = 0$. In the case where we don't need to expand the array, true cost is $\mathcal{O}(1)$ and $\Delta\Phi = 2$ so amortized cost is $\mathcal{O}(1)$. In the case where we do need to expand the array, say from $\ell = n$ to $\ell = kn$, true cost is $\mathcal{O}(n)$ and $\Delta\Phi = 2\Delta n - \Delta\ell = 2 - (k-1)n$, so the amortized cost is $\mathcal{O}(n + (1-k)n) = \mathcal{O}((2-k)n)$. If $k \geq 2$ the amortized cost is $\mathcal{O}(1)$, and if $k < 2$ the amortized cost is $\mathcal{O}(n)$."*

**[Exercise 5 in Lecturer's handout]**

**Exercise 6.C.2** Consider the dynamic array from section 7.3 in lecture notes.

(a) What is wrong with the following argument?

*"Suppose the array starts with $n = 2m$ items, and is at capacity. A single append will require copying these $n$ items. The fundamental rule of amortized analysis is that the true cost of any sequence of operations is $\leq$ the sum of their amortized costs. Since a sequence consisting of a single append can cost $\Omega(n)$, it follows that the amortized cost of append is $\Omega(n)$."*

**[Exercise 6 in Lecturer's handout]**

(b) How can you redefine the potential function so that the analysis is valid?

**Exercise 6.C.3** Consider the dynamic array from section 7.3 in lecture notes, but suppose that when the array needs to be expanded we add a constant $k \geq 1$ to capacity rather than multiplying.

(a) Show that the aggregate cost of appending $n$ items is $\Omega(n^2)$.

(b) An engineer friend tells you excitedly that they have found a cunning potential function that proves that the amortized cost of appending an item is $\mathcal{O}(1)$. Explain why your friend must be mistaken.

(c) Your friend gives the following proof. Where is the flaw?

*"We will add a term to the potential function, measuring how far each item is from the tail end of the array. Specifically, for a dynamic array with $n$ items indexed by $0 \leq i < n$, and capacity $\ell \geq n$, let the potential be*

$$\Phi = 2n - \frac{1}{k}\sum_{i=0}^{n-1}(\ell - 1).$$

*"For example, with $k = 2$, a dynamic array holding $n = 2$ items with capacity $\ell = 4$ has potential $\Phi = 2 \times 2 - (4+3)/2 = 1/2$. When we add an item without expanding capacity, $\Delta\Phi \leq 2$ so appending is $\mathcal{O}(1)$. When we expand capacity, the cost of copying $n$ items is $\mathcal{O}(n)$, and $\Phi$ decreases by $nk/k = n$ because $\ell$ has increased by $k$; hence expanding capacity is $\mathcal{O}(1)$."*

**[Exercise 7 in Lecturer's handout]**

## Fibonacci Heaps

**Recommended reading:**

- CLRS Chapter 19

**Exercise 6.C.4 [Fibonacci Heaps]**

(a) What do the Fibonacci heaps do better than binomial heaps and in what sense?

(b) Give an outline for how Fibonacci heaps work.

(c) Give an outline for proving the guarantees for Fibonacci heaps.

(d) What are the implications for finding shortest paths and MSTs?

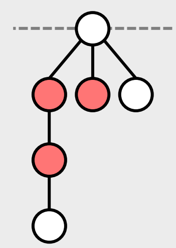
**Exercise 6.C.5** Consider a binary heap. Find a sequence of $N$ items such that inserting them all takes $\Omega(N \log N)$ elementary operations.

**[Exercise 1 in Lecturer's handout]**

**Exercise 6.C.6** An engineer friend tells you excitedly that they have been studying the binary heap and they have found a cunning potential function that proves that the amortized costs of `push` and `popmin` are $\mathcal{O}(1)$ and $\mathcal{O}(\log N)$ respectively. (The big-$\mathcal{O}$ expressions are asymptotic in $N$, the number of items in the heap.)
Given $N$, construct a sequence comprising $m_1$ push and $m_2$ `popmin` operations, for which the heap size never exceeds $N$ and for which the total cost is $\Omega((m_1 + m_2) \log N)$.
Hence explain carefully why your friend is mistaken. [***Hint:*** *You can choose $m_1$ and $m_2$ freely, and they are allowed to depend on $N$. Use your answer to Exercise 5.*]

**[Exercise 15 in Lecturer's handout]**

**Exercise 6.C.7** Give a sequence of operations that would result in a Fibonacci heap of this shape. (The three darker nodes are losers.) What is the shortest sequence of operations you can find?



**[Exercise 10 in Lecturer's handout]**

**Exercise 6.C.8** Prove the result from Section 7.8 of the handout, namely, that in a Fibonacci heap with $n$ items the maximum degree of any node is $\mathcal{O}(n \log n)$. Must it be a root node that has maximum degree?

**[Exercise 11 in Lecturer's handout]**

**Exercise 6.C.9** Consider a Fibonacci heap on which we have only performed `push()` and `popmin()`. Show that, after the cleanup part of `popmin`, the heap has the form of a binomial heap.

**[Exercise 17 in Lecturer's handout]**

**Exercise 6.C.10** In a Fibonacci heap, can a node $x$ acquire a child node $y$, then lose it, then gain it again? [*In the complexity analysis of the Fibonacci heap, we carefully wrote "children ... in the order of when they last became children of $x$." This question explains why we needed the word 'last'.*]

**[Exercise 18 in Lecturer's handout]**

**Exercise 6.C.11** In the Fibonacci heap, suppose that `decreasekey()` only cuts out (if necessary) the node whose key has been decreased, i.e. it doesn't use the loser flag and it doesn't recursively inspect the node's parents. Show that it would be possible for a node with $n$ descendants to have $\Omega(n)$ children.

**[Exercise 19 in Lecturer's handout]**

## Disjoint sets

**Recommended reading:**

- CLRS Chapter 21
- Jeff Errickson's notes

**Exercise 6.C.12 [Disjoint Sets]**

(a) Which operations does the disjoint set data structure support?

(b) How does it support each of the operations?

(c) What is the time complexity for each of the operations?

(d) Implement the disjoint set data structure. You may want to test your implementation on **[CSES Road Construction]**.

**Exercise 6.C.13** In the flat forest implementation of disjoint set, using the weighted union heuristic, prove the following:

(a) After an item has had its 'parent' pointer updated $k$ times, it belongs to a set of size $\geq 2^k$.

(b) If the disjoint set has $N$ items, each item has had its 'parent' pointer updated $\mathcal{O}(\log N)$ times.

(c) Starting with an empty disjoint set, and assuming the number of items added is $\leq N$, show that any sequence of $m$ operations takes $\mathcal{O}(m + N \log N)$ time in aggregate.

**[Exercise 13 in Lecturer's handout]**