BLACKBOARD PROOFS

CSE202 – WEEK 9

1. Amortized complexity of a sequence of `append`

When the Nth append is performed, the capacity of the table is larger than N and the copies performed up to that capacity were all of tables of capacity smaller than N. Moreover, increasing the capacity by the lines of code

```
if newsize>self.capacity:
    self.realloc((int)(alpha*newsize))
```

leads to the sequence (t_k) of capacities satisfying the recurrence

$$t_{k+1} = \lfloor \alpha(t_k + 1) \rfloor, \qquad \text{with } \alpha > 1.$$

There, $\lfloor \alpha(t_k + 1) \rfloor$ is a direct translation of the statement `(int)(alpha*newsize)` and the fact that `newsize`, the argument of `resize`, takes for values successive integers, so that the first time it is larger than the capacity t_k is when it equals $t_k + 1$.

Thus, we need to compute an upper bound for

$$S_N := \sum_{t_k \le N} t_k.$$

We denote by m the index of the last t_k in the sum, for which we have $t_m \le N$. Using the inequality $x - 1 < \lfloor x \rfloor$, the recurrence gives

$$\alpha(t_{k-1} + 1) - 1 < t_k$$

whence, since $\alpha > 1$,

$$t_{k-1} < \frac{1}{\alpha}(t_k + 1) - 1 < \frac{1}{\alpha} t_k$$

and finally,

$$S_n = t_m + t_{m-1} + t_{m-2} + \cdots \le N \left(1 + \frac{1}{\alpha} + \frac{1}{\alpha}^2 + \cdots \right) = \frac{\alpha}{\alpha - 1}.$$

2. Amortized complexity of a sequence of `append` and `delete` by the accounting method

We consider a sequence of N operations of type `append` or `delete`, with the algorithm that changes the capacity from N to αN when the table is full and from N to $\alpha\beta N$ when it has at most βN elements (and necessarily $\alpha > 1$ to enable growth and $\alpha\beta < 1$ to allow shrinking).

First, observe that after a resizing of the table, following an insertion or a deletion, its filling ratio is always $1/\alpha$. If N denotes the current capacity, the cost that should be charged to each `append` operation is as before 1 for the element itself, 1 for its future copy and a cost for the future copy of the N/α elements that are already there in the table, divided by the minimal number of elements that will be appended before the next copy due to an `append`, which is $N(1-1/\alpha)$. This gives a charge

$$1+1+\frac{N/\alpha}{N(1-1/\alpha)}=1+\frac{\alpha}{1-\alpha}.$$

Similary, the cost that should be charged to each `delete` operation is 1 for its deletion itself, plus a cost for the future copy of the βN elements, divided by the minimal number of elements that will be deleted before that next copy due to a `delete`, which is $(1/\alpha-\beta)N$. This gives a charge

$$1+\frac{\beta N}{(1/\alpha-\beta)N}=1+\frac{\alpha\beta}{1-\alpha\beta}.$$

Thus we have established that the amortized cost of a sequence of N `append` or `delete` is $O(N)$.

3. Properties of the rank

Here is a detailed proof that the three assertions on slide 13 still hold in the case where the path compression method is used. The proof also applies in the case when path compression is not used.

A first observation is that rank changes occur only at the root of a tree and only during a link operation.

The rank increases from leaf to root. The proof is by induction on the steps of the algorithm. The property holds initially when all trees are reduced to one node of rank 0. The only changes in a tree are by compression or link.

In a compression, all nodes below the root are grafted to the root, which, by induction hypothesis, has a higher rank. Thus the property is preserved by compression.

For edges that are unaffected by a link operation, the property is preserved. If the root labeled is unchanged, then the only modified edge is the new edge to the second tree, whose root has a lower rank, so that the property is preserved. If both roots had the same rank before the link, then one of them sees its rank increased, so that it is still larger than that of its former children, and it becomes larger than that of its new child.

The size of a subtree is at least $2^{\text{rank(root)}}$. Again, by induction on the steps of the algorithms.

Initially, all trees have size 1 and a root of rank 0; the property holds.

In a compress operation, neither size nor rank of the root is changed, so that the property is preserved.

Consider now two trees whose roots have rank r_1 and r_2. By the induction hypothesis, they have subtrees with at least 2^{r_1} and 2^{r_2} nodes. If $r_1\neq r_2$, then the link operation does not change any of the ranks; the size of the subtree of the root with the smaller rank is unchanged and the other one is increased, so that the property is preserved. If $r_1=r_2$ then one of the nodes becomes the child of the other one and keeps its rank and subtree. The other one gets rank r_1+1 and its subtree has size at least $2^{r_1}+2^{r_2}=2^{r_1+1}$, which concludes the proof.

The number of nodes or rank r is $\leq n/2^r$. By the previous statement, each node of rank r has a subtree of size at least 2^r. Thus the total number of nodes of that rank is at most $n/2^r$.

The number of nodes of rank $> s$ is $\leq n/2^s$. This is a consequence of the previous one that is used on slide 18. It is obtained by summing the previous bound for $r = s+1, s+2, \ldots$.