PSAT Week 5: Sorting Algorithms II

Objectives: Comparison complexity, Merge Sort, Divide&Conquer principle

Recapitulation:

- Selection Sort: Repeatedly find the smallest card of remaining pile
- Complexity: Number of elementary instructions

Comparison Complexity of Selection Sort:

→ Number of comparisons for minimum of k cards: k-1
→ Selection Sort for pile of n cards: Minimum with k ranging from n to 1
→ The total number of comparisons is at most:

```
 n-1 + n-2 + n-3 + … + 2 + 1 =                  (double counting)
(n-1 + n-2 + n-3 + … + 2 + 1 +
 n-1 + n-2 + n-3 + … + 2 + 1) / 2 =             (rearranging )
(1   + 2   + 3   + … + n-3 + n-2 + n-1 +
 n-1 + n-2 + n-3 + … + 3   + 2   + 1) / 2 =     ( rearranging )
[(1 + n-1) + (2 + n-2) + (3 + n-3) + … + (n-1 + 1)] / 2 =
[n         + n         + n         + … + n        ] / 2 =
n*(n-1)/2
```

Task 1 (10 min):

- Develop an algorithm that, given two sorted piles of cards, creates a single sorted pile containing all given cards.
- Naively, one could simply join the two piles and perform Insertion Sort, but a more efficient solution is possible.

Merge Procedure:

```
Put the first given pile to the left and the second given pile to the
right and turn both such that the front side of the cards faces upwards
Start an empty pile in the middle
Repeat the following instructions until the left pile or the right pile is
empty:
      Compare top cards of left pile and right pile and select smaller one
      Put selected card on middle pile such that front side of the cards
      faces downwards
If the left pile or the right pile is not yet empty, put the whole
remaining pile on the middle pile with the front side of the cards facing
downwards
Return the middle pile as the result
```

Analysis of Merge Procedure:

- After every comparison the middle pile grows by one card
- In the end, the middle pile contains all cards
- If the total number of cards is k, there can thus be most k comparisons

Merge Sort Algorithm:

```
If the pile consists of a single card, do nothing and return this card as
the result
Otherwise, perform the following instructions:
      Divide the given pile into two halves that differ in size by at most
      one card
      Sort the two piles independently (using Merge Sort!)
      Perform the Merge Procedure on the two sorted piles
      Return the result of the Merge Procedure
```

Task 2 (5 min)

- Execute the Merge Sort algorithm on 16 cards.
- Track the number of pairwise comparisons of cards you perform.

Divide&Conquer Principle:

- **Divide&Conquer**: Divide the problem into several smaller subproblems and combine the solutions to the subproblems to a solution for the original problem
- **Recursion:** Algorithm calls itself
- Divide&Conquer algorithms often use recursion, but could also call other algorithms to solve the subproblems

Task 3 (20 min):

- How many comparisons does Merge Sort perform at most for 4, 8, 16 cards?
- Can you give a general upper bound on the number of comparisons? (You may assume that the number of cards is a power of two, i.e., $n=2^i$)

Analysis of Merge Sort (15 min):

- Analysis tool: recursion tree

n	1 merge with n cards → n comparisons
n/2 n/2	2 merges with n/2 cards → ≤2*n/2=n comparisons
n/4 n/4 n/4 n/4	4 merges with n/4 cards → ≤4*n/4=n comparisons
…	
2 2 2 2 … 2 2 2	n/2 merges with 2 cards → ≤n/2*2=n comparisons

- At each level: n comparisons
- How many levels are there?
- First level: n cards, last level 2 cards; number of cards halved at each level
- Backward analysis: $2^{\#levels} = n \rightarrow \#levels = \log_2(n)$
- Thus, there $n*\log_2(n)$ comparisons