ECE 2400 Computer Systems Programming

Topic 9: Sorting Algorithms

`http://www.csl.cornell.edu/courses/ece2400`
School of Electrical and Computer Engineering
Cornell University

revision: 2025-03-05-00-59

Please do not ask for solutions. Students should compare their solutions to solutions from their fellow students, discuss their solutions with the instructors during lab/office hours, and/or post their solutions on Ed for discussion.

List of Problems

Problem 1. Short Answer

Carefully plan your solution before starting to write your response. Please be brief and to the point; if at all possible, limit your answers to the space provided.

Part 1.A 3-Way Merge Sort

Consider the following variant of merge sort. The `merge_sort_h` helper function will perform a merge sort on the given array `x` in the range of the indices [`begin`,`end`). In other words, `begin` is the minimum index and `end` is the maximum index (exclusive). Note that this algorithm is also using a `merge3` helper function which takes as parameters a destination array and three sorted input arrays represented with begin/end indices. You can assume this helper function is implemented in a similar way as how we merged two partitions in lecture. We are also assuming a slightly different interface for the insertion sort which takes an input array and begin/end indices into that array. **What is the worst-case time complexity of this algorithm as a function of N. Use asymptotic big-O notation. Use the space on the next page to justify your answer.** While we encourage you to think through the six-step process described in lecture, you are not required to explicitly show each step. A simpler high-level argument will probably be sufficient. *We recommend drawing a picture as part of your justification.*

```
void merge3_sort_h( int* x, int first, int last )
{
  int size = last - first;
  if ( size <= 4 ) {
    insertion_sort( x, first, last );
    return;
  }

  int mid1 = first +   ( size / 3 );
  int mid2 = last  + 2*( size / 3 ) + 1;

  merge3_sort_h( arr, first, mid1 );
  merge3_sort_h( arr, mid1,  mid2 );
  merge3_sort_h( arr, mid2,  last );

  int* tmp = malloc( size * sizeof(int) );
  merge3( tmp, x, first, mid1, x, mid1, mid2, x, mid2, last );

  int j = 0;
  for ( int i = first; i < last; i++ ) {
    x[i] = tmp[j];
    j += 1;
  }

  free(tmp);
}

void merge3_sort( int* x, int n )
{
  merge3_sort_h( x, 0, n );
}
```