Due: Thursday, September 24

Read Sections 2.1, 2.2, 2.3, 2.4, 2.5, 2.7, 1.1, 1.2

Do the exercises 1.5, 1.6, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.11, 2.14, 2.19 from GKP. Do not submit your solutions, but check them with the answers from the back of the text.

1. (9 points) Study Exercise 2.20 in the text carefully. Find a closed form for the value of
   \[ \sum_{n \geq 0} kH_k. \]

2. (3 points) Consider the following function which returns the largest and the smallest elements of a set \( A \).
   \[
   \text{function MaxMin}(A)
   \begin{align*}
   & \text{if } |A|=1 \text{ (let } A=\{a\} \text{) then return } (a,a) \\
   & \text{if } |A|=2 \text{ (let } A=\{a,b\} \text{) then return } (\text{Max}(a,b), \text{Min}(a,b))
   \end{align*}
   \]
   Partition \( A \) into \( A_1 \) and \( A_2 \) such that \( |A_1| = \lfloor |A|/2 \rfloor \) and \( |A_2| = \lceil |A|/2 \rceil \)
   \( (\text{big1},\text{little1}) \leftarrow \text{MaxMin}(A_1) \)
   \( (\text{big2},\text{little2}) \leftarrow \text{MaxMin}(A_2) \)
   \text{return } (\text{Max}(\text{big1},\text{big2}), \text{Min}(\text{little1},\text{little2}))
   \]
   Let \( f(n) \) be the number of invocations of \text{Max} and \text{Min} when \text{MaxMin} is called on a set of \( n \) elements.
   (A) Find a recurrence for \( f(n) \), assuming that \( n \) is a power of 2.
   (B) Solve the recurrence for \( f(n) \), assuming that \( n \) is a power of 2.

3. (5 points) Some sorting algorithms (local transposition sorts) function by swapping adjacent pairs of elements which are in the wrong order. The analysis of these algorithms is very related to the concept of inversions. If \( x_1,\ldots,x_n \) is a permutation of \( 1,\ldots,n \), then an inversion is a pair \( 1 \leq i < j \leq n \) if \( x_i > x_j \). For example, the permutation 3,1,2,4 has two inversions, (1,2) and (1,3).
   (A) Define a recurrence for \( I_{\text{max}}(n) \), the maximum number of inversions of a sequence of \( n \) elements.
   (B) For \( n \geq 1 \), how many permutations of \( 1,\ldots,n \) have \( I_{\text{max}}(n) \) inversions?
   (C) Define a recurrence for \( I_{\text{min}}(n) \), the minimum number of inversions of a sequence of \( n \) elements.
   (D) For \( n \geq 1 \), how many permutations of \( 1,\ldots,n \) have \( I_{\text{min}}(n) \) inversions?
   (E) Find closed form solutions for \( I_{\text{max}}(n) \) and \( I_{\text{min}}(n) \).

4. (8 points) A graph is a set \( V=\{1,\ldots,n\} \) of vertices and a set \( E \) of unordered pairs of vertices which are the edges. For any \( S \subseteq V \), we let graph \( G-S \) denote the graph obtained from \( G \) by removing \( S \) and all edges incident with \( S \). Associated with a graph is a neighborhood function, \( \Gamma : V \rightarrow 2^V \), satisfying
\( \Gamma(v)=\{v\} \cup \{w| (v, w) \in E}\). That is, for any vertex \( v \in V \), \( \Gamma(v) \) is the set containing \( v \) and all vertices adjacent to \( v \). A set of vertices \( S \subseteq V \) is independent if the vertices are pairwise nonadjacent, that is, if there is no edge between any two vertices of \( S \). The independence number of a graph \( G \), \( \alpha(G) \), is the cardinality of a largest independent set of vertices of \( G \). For example, for graph \( G \)

\[
V=\{1,2,3,4,5,6\}, E=\{\{1,4\},\{1,5\},\{2,5\},\{3,4\},\{4,6\}\}, \Gamma(1)=\{1,4,5\}, \Gamma(2)=\{2,5\}, \text{ the sets } \{5\}, \{3,5\}, \{1,2,3,6\} \text{ and } \{1,2\} \text{ are all independent sets, and } \alpha(G)=4. \text{ The graph } G-\{1,4\} \text{ is}
\]

We are going to start to analyze two algorithms for computing \( \alpha(G) \).

A) For any graph \( G \) and any vertex \( v \) of \( G \), an independent set of \( G \) either doesn't contain \( v \) or it does. If an independent set of \( G \) contains \( v \), then it doesn't contain any other vertex of \( \Gamma(v) \).

function \( \alpha_1(G) \)

if \( E=\emptyset \)

then return \( |V| \)

else choose some \( v \) of \( G \) such that \( \Gamma(v) \geq 2 \)

return max \( \{ \alpha_1(G-\{v\}), 1+\alpha_1(G-\Gamma(v)) \} \)

Assume that if \( G \) contains \( n \) vertices, each invocation of \( \alpha_1(G) \) requires \( n \) operations, plus the time to call \( \alpha_1(G-\{v\}) \) and \( 1+\alpha_1(G-\Gamma(v)) \). Develop a recurrence for the worst-case time to compute \( \alpha_1(G) \). You needn’t solve the recurrence. Describe a graph which forces \( \alpha_1(G) \) to use this worst-case time.

B) We could speed up Algorithm \( \alpha_1(G) \) by choosing vertex \( v \) with large \( \Gamma(v) \).

function \( \alpha_2(G) \)
if $\Gamma(v) \leq 2$ for each $v \in V$

then return $|V| - |E|$

else choose some $v$ of $G$ such that $\Gamma(v) \geq 3$

return $\max \{ \alpha_2(G - \{v\}), 1 + \alpha_2(G - \Gamma(v)) \}$

Assume that if $G$ contains $n$ vertices, each invocation of $\alpha_2(G)$ requires $n$ operations, plus the time to call $\alpha_2(G - \{v\})$ and $1 + \alpha_2(G - \Gamma(v))$. Develop a recurrence for the worst-case time to compute $\alpha_2(G)$. You needn’t solve the recurrence.