1. (20 marks) For each of the following functions:
   
   • (4 marks) Rank the function relative to the others in order of increasing growth rate (slowest to fastest). No justification is required.
   
   • (8 marks) Classify into $\Theta(1)$, $\log^{\Theta(1)}(n)$, $n^{\Theta(1)}$, $\theta(n^2)$ or $2^{\Omega(n)}$. No justification required.
   
   • (8 marks) Approximate the sum $\Theta(\sum_{i=1}^{n} f(i))$. Specify the rule or method you used.

(a) $f(n) = n^2(\log n)^2$
   
i. Rank:
      • Answer: 2
   
   ii. Class:
      • Answer: $n^{\Theta(1)}$
   
   iii. Sum:
      • Answer: $n^3(\log n)^2$, since $f(n)$ is arithmetic-like.

(b) $f(n) = (\log n)^n$
   
i. Rank:
      • Answer: 4
   
   ii. Class:
      • Answer: $2^{\Omega(n)}$
   
   iii. Sum:
      • Answer: $(\log n)^n$, since $f(n)$ is geometric-like.

(c) $f(n) = n^2 \log (n^2)$
   
i. Rank:
      • Answer: 1
   
   ii. Class:
      • Answer: $n^{\Theta(1)}$
   
   iii. Sum:
      • Answer: $n^3 \log (n^2) = \Theta(n^3 \log n)$, since $f(n)$ is arithmetic-like.

(d) $f(n) = \left(\frac{3}{2}\right)^n$
   
i. Rank:
      • Answer: 3
   
   ii. Class:
      • Answer: $2^{\Omega(n)}$
   
   iii. Sum:
      • Answer: $\left(\frac{3}{2}\right)^n$, since $f(n)$ is geometric-like.
2. (10 marks) Solve the following recurrence relations. Justify your answer.

(a) (5 marks) \( T(n) = 2T(n/2) + n + \log n \)

- Answer: \( T(n) \) has the form \( T(n) = a \log(n/b) + f(n) \), where \( a = b = 2 \) and \( f(n) = n + \log n \). Since \( f(n) \in \Theta(n^{\log_b a}) = \Theta(n) \), \( T(n) \in \Theta(n \log n) \), by the Master Theorem.

(b) (5 marks) \( T(n) = 2T(n/2) + n \log n \)

- Answer: Again, \( T(n) \) has the form \( T(n) = a \log(n/b) + f(n) \), where \( a = b = 2 \) and \( f(n) = n \log n \). Here, however, the Master Theorem does not apply, since \( f(n) = n \log n \notin \Theta(n^{1+\epsilon}) \), for any \( \epsilon \geq 0 \). Instead, we use the recursion tree method:

<table>
<thead>
<tr>
<th>Level</th>
<th>Instance Size</th>
<th>Work in Stackframe</th>
<th>Number of Stackframes</th>
<th>Work in Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( n )</td>
<td>( n \log n )</td>
<td>1</td>
<td>( n \log n )</td>
</tr>
<tr>
<td>1</td>
<td>( n/2 )</td>
<td>( (n/2) \log(n/2) )</td>
<td>2</td>
<td>( n \log(n/2) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\log n</td>
<td>( n/2^{\log n} = 1 )</td>
<td>( (n/2^n) \log(n/2^n) = 0 )</td>
<td>( 2^{\log n} = n )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

Thus the total work \( T(n) \) is given by

\[
T(n) = \sum_{i=0}^{\log n} n \log (n/2^i) = n \sum_{i=0}^{\log n} (\log n - i) = n \sum_{i=0}^{\log n} i \in \Theta(n \log^2 n)
\]

3. (20 marks) Proving Correctness

(a) (10 marks) What are the steps required to prove the correctness of an iterative algorithm? Be brief.

- Answer:
  i. Given any input satisfying the pre-condition, and following any pre-code, the loop invariant is established.
  ii. Given that the loop invariant is true at iteration \( i \), execution of the loop-code will maintain the loop invariant as true at iteration \( i + 1 \).
  iii. The loop invariant, combined with the exit condition and any post-code, establishes the post-condition.
  iv. By the measure of progress, the loop starts a finite distance from the exit condition and this distance is steadily reduced.

(b) (10 marks) What are the steps required to prove the correctness of a recursive algorithm? Be brief.

- Answer:
  i. Recursive calls are made with smaller input that satisfies the pre-condition.
  ii. General case: Assuming the algorithm is correct for valid inputs smaller than \( n \), the algorithm will establish the post-condition for any input of size \( n \) meeting the pre-condition.
  iii. Base case: For any input that meets the pre-condition and is less than some specified size, the algorithm establishes the post-condition directly, with no recursive calls.
4. (50 marks) Convex hulls

Given a set of $n$ points $p_i \in P$ in the plane, the convex hull $C$ of $P$ is the smallest convex polygon containing $P$. The convex hull can be specified as the clockwise cycle of vertices of the polygon. For simplicity, you can assume that the points are in general position, i.e., no 3 points are collinear. Fig. 1 shows an example of the convex hull of 13 points.

![Figure 1: A convex hull](image)

It can be shown that every vertex of the convex hull is a point in $P$. We are interested in the problem of computing the convex hull.

(a) (3 marks) Prove that the leftmost point in $P$ must be a vertex in $C$. Be brief.

- **Answer:** Let $p_l$ be the leftmost point of $P$. $p_l$ cannot be outside $C$, or $C$ would not be a convex hull. If $p_l$ is interior to $C$, there must be a vertex of $C$ to the left of $p_l$, contradicting the assumption that $p_l$ is the leftmost point of $P$. Thus $p_l$ must lie on the boundary of $C$. If $p_l$ is not a vertex of $C$, it must lie on a vertical line segment of $C$, since no vertex can be to the left of $p_l$. But this is impossible, since no 3 points of $P$ are collinear. Thus $p_l$ is a vertex of $C$.

(b) (3 marks) Let function $\text{Right}(P, p_i, p_j)$, $p_i, p_j \in P$ return true if and only if all other points in $P$ lie to the right of the line through $(p_i, p_j)$. Prove that if $\text{Right}(P, p_i, p_j)$, then $(p_i, p_j)$ is part of the clockwise convex hull.

- **Answer:** The proof is similar to the proof for the leftmost point of $P$. $p_i$ and $p_j$ cannot be outside $C$, or $C$ would not be a convex hull. If $p_i$ or $p_j$ is interior to $C$, there must be a vertex of $C$ to the left of $(p_i, p_j)$, contradicting $\text{Right}(P, p_i, p_j)$. Thus $p_i$ and $p_j$ must lie on the boundary of $C$. Since there are no vertices to the left of $(p_i, p_j)$, the line segment $(p_i, p_j)$ must form part of the boundary of $C$. Since no 3 points in $P$ can be collinear, $p_i$ and $p_j$ must be the only points in $P$ on this line, and therefore must be vertices of $C$.

(c) (3 marks) Assuming $\text{Right}$ takes $\Theta(n)$ time, prove that the problem must be $\mathcal{O}(n^3)$

- **Answer:** One algorithm for computing $C$ is to consider all $n(n-1)$ lines through all pairs of points $(p_i, p_j)$. Those for which $\text{Right}(P, p_i, p_j)$ returns true are in $C$. This will take $\Theta(n^3)$ time. These can then be sorted into the correct cyclical order by first selecting one pair $(p_i, p_j)$ at random and then finding the pair $(p_j, p_k)$, and repeating. Since this sort will take $\mathcal{O}(n^2)$ time, the whole algorithm is $\Theta(n^3)$, and the problem is $\mathcal{O}(n^3)$.

(d) You are now to design an iterative algorithm that uses $\text{Right}$ to solve the problem more efficiently. Your algorithm will be given a set $P$ of 3 or more points in general position (i.e., no 3 points...
Your algorithm will return a sequence $C = \{c_0, c_1, \ldots, c_k\}$ of points forming the vertices of the convex hull. Your algorithm will start by setting $c_0$ to the leftmost point in $P$ (e.g., $p_{12}$ in Fig. 1). It then proceeds clockwise to add points monotonically to $C$, until $c_k = c_0$.

i. (8 marks) Specify your algorithm in concise pseudocode, making use of function $\text{Right}$.

**Answer:**

```algc
algorithm Hull(P)
  <pre-cond>: $P$ is a set of at least 3 points in the plane, in general position
  <post-cond>: Returns the clockwise cycle of vertices on the convex hull of $P$.
  begin
    1 $C[0] = \text{Leftmost}(P)$
    2 $C[1]=n1$
    3 $i = 1$
    4 while $C[i] \neq C[0]$
        5 $j = 0$
        6 while $\neg\text{Right}(P, C[i-1], P[j])$
            7 $j = j + 1$
        8 end
        9 $C[i] = P[j]$
        10 $i = i + 1$
  end
  return $C$
end algorithm
```

ii. (8 marks) Prove the correctness of your algorithm. Make sure that you complete all steps of the proof.

**Answer:** The loop-invariant for the main loop (Line 4) is: $LI: \{c_0, \ldots, c_{i-1}\}$ is a subsequence of $C$. The loop is first entered when $i = 1$, and since $c_0$ is a vertex in $C$, the LI is established. The inner loop finds a point $p_j \in P$ such that $(c_{i-1}, p_j)$ is part of $C$, and so setting $c_i = p_j$ on Line 9 maintains the LI. Note that such a $p_j$ always exists, and $c_i \neq c_j, i \neq j \neq 0$, since otherwise $\{c_i, \ldots, c_j\}$ would form a polygon with $c_0$ on the outside (left). Thus the loop can run at most $n$ times before $c_i = c_0$, and the algorithm halts. When the algorithm halts, $c$ forms a clockwise cycle of vertices such that all points in $P$ lie to the right of each line $(c_i, c_{i+1})$, and thus are inside or on the boundary of the corresponding polygon.

iii. (3 marks) Assuming $\text{Right}$ takes $\Theta(n)$ time, provide a tight bound on the running time $T(n, k)$ of your algorithm, in terms of the number $n$ of points in $P$ and the number $k$ of points in the convex hull. Briefly justify your answer.

**Answer:** The inner loop calls $\text{Right}$ at most $n+1$ times on each iteration of the outer loop, and so the running time is $\Theta(kn^2)$.

iv. (3 marks) Provide a tight bound on the running time $T(n)$ of your algorithm in terms of $n$ only. What kind of configuration of points would form a worst case?

**Answer:** If all points in $P$ are on the convex hull, then $k = n$ and the running time is $\Theta(n^3)$.

**Note:** This is almost the “giftwrapping” algorithm due to Jarvis. Instead of using $\text{Right}$, we can directly determine the point $p_j$ such that $(p_{j-1}, p_j)$ has minimum clockwise angle with respect to the vertical. This angle can be determined in $\Theta(n)$ time, leading to a running time of $\Theta(n^2)$.

(e) Now consider a different approach. Let $P_1$ and $P_2$ be disjoint subsets of $P$ such that $P_1 \cup P_2 = P$ and all points in $P_1$ are to the left of all points in $P_2$. Suppose that you have a function $\text{Mergehulls}(C_1, C_2)$ which, given the convex hulls for $P_1$ and $P_2$, can return the convex hull for $P$ in $\Theta(n)$ time.


i. (8 marks) Design a recursive algorithm based upon this function. Specify your algorithm using concise pseudocode.

**Answer:** The points in $P$ are first sorted by their $x$–coordinates, and then passed to the algorithm below, which returns the convex hull.

**algorithm Hull($P$)**

<**pre-cond**: $P$ is a set of points in the plane, sorted by their $x$–coordinates. The points are in general position, and $|P| \geq 1$.

<**post-cond**: Returns the cycle of vertices on the convex hull of $P$, or $P$ if $|P| < 3$.>

begin
1 Let $n = |P|
2$ if $n \leq 3$
3 return $P$
4 end
5 Let $m = \lfloor n/2 \rfloor$
6 if $P[m].x = P[m-1].x$
7 $C_1 = \text{Hull}(P[0 \ldots m])$
8 $C_2 = \text{Hull}(P[m+1 \ldots n-1])$
9 else
10 $C_1 = \text{Hull}(P[0 \ldots m-1])$
11 $C_2 = \text{Hull}(P[m \ldots n-1])$
12 end
13 $C = \text{Mergehulls}(C_1, C_2)$
14 return $C$
end algorithm

ii. (8 marks) Prove the correctness of your algorithm (assuming $\text{Mergehulls}$ is correct). Make sure that you complete all steps of the proof.

- **Answer:** If $n = 3$, all points in $P$ are on the convex hull, and so the postcondition is satisfied by returning $P$ directly. If $n > 3$, we partition the points into left and right halves. Note that recursive calls receive smaller input. Note also that partitioning will produce sets with at least 1 point, since inputs with fewer than 4 points are handled directly. (We assume that $\text{Mergehulls}$ handles inputs of size 1 or 2 appropriately.) Since no 3 points are collinear, at most 2 points can have the median $x$ value, and by ensuring both are in the same set, we ensure that all points in the first partition are to the left of all points in the second. Thus the precondition for $\text{Mergehulls}$ is met and $C$ is the convex hull of $P$.

iii. (3 marks) Provide a tight bound on the running time $T(n)$ of your algorithm in terms of $n$. Justify your answer.

- **Answer:** Sorting takes $\Theta(n \log n)$ time. The running time of $\text{Hull}$ is given by $T(n) = 2T(n/2) + cn$, since $\text{Mergehulls}$ is $\Theta(n)$, so that $\text{Hull}$ also takes $\Theta(n \log n)$ time. Thus the total running time of the algorithm is $\Theta(n \log n)$. 