Skip List Analysis

Let $P$ be the search path for some key $k$. Let $v$ be some node in the search path. Prove that node $v$ is the highest node in its tower if and only if the search path $P$ came to $v$ from the left.

Interpolation Search

Construct an array $A$ such that interpolation search for some key $K$ takes $\Theta(n)$ time. Specify what $K$ is. Here is the code for interpolation search from the slides for reference:

```plaintext
Interpolation-search(A, n, k)
A: Array of size n, k: key
1. $\ell \leftarrow 0$
2. $r \leftarrow n - 1$
3. while $(\ell < r) \& \& (A[r] = A[\ell]) \& \& (k \geq A[\ell]) \& \& (k \leq A[r])$
4. $m \leftarrow \ell + \lfloor \frac{k-A[\ell]}{A[r]-A[\ell]} \cdot (r-\ell) \rfloor$
5. if $A[m] < k$ $\ell \leftarrow m + 1$
6. elseif $A[m] = k$ return $m$
7. else $r \leftarrow m - 1$
8. if $(k = A[\ell])$ return $\ell$
9. else return “not found, but would be between $\ell - 1$ and $\ell$”
```

MTF

We saw MTF in class. If we are using lists, then Insert with MTF-heuristic can be done in constant time. How can we implement MTF on a dynamic array so that Insert takes constant time?
Convex Hull

Recall: A convex set is a set of points such that, given any two points \( a \) and \( b \) in that set, the line joining \( a \) and \( b \) lies entirely within that set.

The convex hull of a set of \( n \) points in the plane is the smallest convex set that contains all points. Below is a figure of a set of points and another figure showing a red outline containing all points forming the convex hull for that set.

Note: Understanding the concept of a convex hull is NOT important to the question

Tim has designed an algorithm that gets as input \( n \) points and a parameter \( H \), and it outputs, in \( O(n \log H) \) time, one of the following:

- “This \( H \) is too small”, or
- “Here is the convex hull, and it contains \( h \) points for \( h \leq H \)”

Design an algorithm that finds the convex hull in \( O(n \log h) \) time, where \( h \) is the number of points in the convex hull.

If you are curious about the convex hull finding algorithm, you can read up on it more here: Chan’s algorithm