Convex Hull of a Simple Polygon in Linear Time

This note concerns the computation of the convex hull of a given simple polygon. Suppose the sequence of vertices of the given polygon P is $p_1, p_2, \cdots, p_n$. One way to solve the problem is to ignore the fact that the sequence forms a simple polygon and just find the convex hull of the vertices of P. This would take $O(nlgn)$ time, say by the Graham scan method. Section 4.1.4 of Preparata-Shamos [PrS85] describes an $O(n)$ time algorithm to compute $conv(P)$. Below, we will discribe a simpler linear time algorithm due to Melkman [1987]:

A. Melkman, *"On-line construction of the convex hull of a simple polyline"*, Information Processing Letters, vol. 25, pp. 11-12, 1987.

This algorithm is also capable of finding the convex hull of a simple polyline (an open nonself-intersecting chain of line segments).

We will consider the vertices in the given order and use an incremental approach very similar to the second phase of the Graham scan algorithm to construct the convex hull. Suppose we already have the convex hull of $\{ p_1, p_2, \cdots, p_{i-1} \}$. How should we update it to get the convex hull of $\{ p_1, p_2, \cdots, p_{i-1}, p_i \}$? We will use a *deque* (double ended queue) D to represent the sequence of vertices on the convex hull of $\{ p_1, \cdots, p_i \}$. The deque can be implemented by a sequential list (e.g. a circular array or a circular linked list). Furthermore, assume b and t point to the bottom and the top of D, respectively. We shall denote the sequence of vertices in D from bottom to top as $v_b, v_{b+1}, \cdots, v_{t-1}, v_t$. We use the 4 primitive deque operations $PopBottom(D)$, $PopTop(D)$, $PushBottom(p, D)$, and $PushTop(p, D)$ with the obvious meanings. For instance $PushBottom(p, D)$ means add point p to the bottom of deque D and appropriately update its bottom pointer. We will make the convention that the deque sequence gives the *closed* boundary of the convex hull of the points considered so far. That is, $v_b = v_t$. Now the algorithm is as follows:

Algorithm $Convex - Hull\ (\ P\ )$;
  $D \leftarrow (\ p_2, p_1, p_2\ )$; /* i.e., convex-hull of $\{ p_1, p_2 \}$ */
  for $i \leftarrow 3$ **to** n **do**
    if p_i is outside the angle $v_{t-1} v_t v_{b+1}$ **then do**
      while p_i is left of $\overrightarrow{v_b v_{b+1}}$ **do** $PopBottom(D)$;
      while p_i is right of $\overrightarrow{v_t v_{t-1}}$ **do** $PopTop(D)$;
      $PushBottom(p_i, D)$; $PushTop(p_i, D)$
    end
  output D /* vertices of $conv(P)$ in order around the convex hull boundary */
end

We now sketch a proof of correctness. We first consider the case in which p_i is discarded. This happens when p_i is inside the angle $v_{t-1} v_t v_{b+1}$. (See Figure 1(a).) We know that v_{b+1} is connected to v_{t-1} by a polygonal path, and that p_i is connected to v_b by a

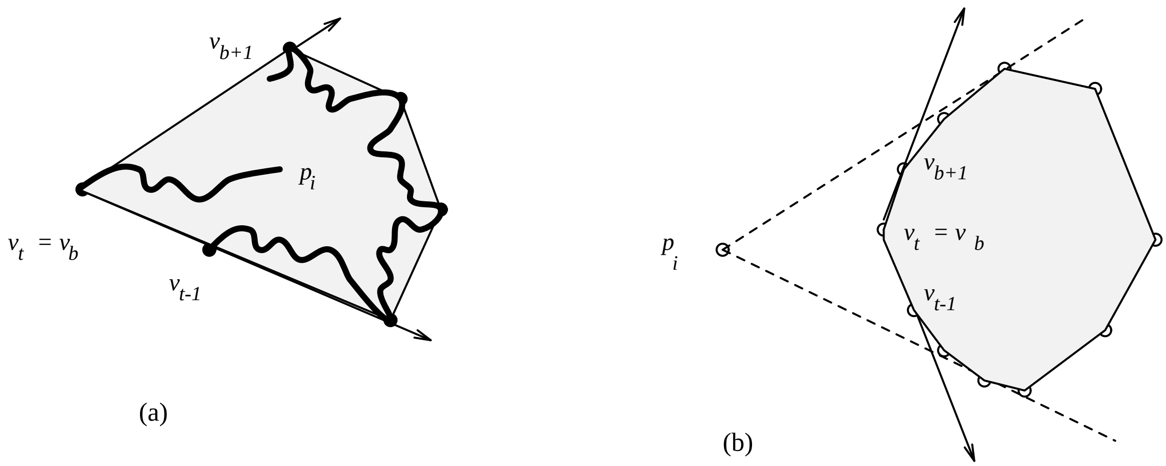

Figure 1. The two cases for the position of p_i.

polygonal path. The two paths do not intersect, so p_i must lie inside the current convex-hull. When p_i is not discarded, it lies outside the current hull, and the algorithm pops hull vertices until it gets to the endpoints of the tangents from p_i to the current hull. (See Figure 1(b).) The algorithm is linear: if it operates on an n–vertex polygon, it does at most $2n$ pushes and $2n - 3$ pops.