Shortcutting lists

Michiel Smid*

June 25, 2003

Logarithms are binary. We define $\log 0 := 0$.

1 Shortcutting lists

Let $V := \{x_1, x_2, \ldots, x_n\}$ be a set of n vertices and let L be the undirected graph on V with edge set $\{\{x_i, x_{i+1}\} : 1 \leq i < n\}$. Hence, L is the list containing the vertices of V in increasing order of their indices. We will write this list as $L = (x_1, x_2, \ldots, x_n)$. We will consider undirected graphs that contain L. Let $G = (V, E)$ be such a graph and let i and j be any two indices such that $1 \leq i \leq j \leq n$. We will say that x_i is *to the left of* x_j in L. A path

$$P = (x_i = x_{i_0}, x_{i_1}, x_{i_2}, \ldots, x_{i_k} = x_j)$$

in G between x_i and x_j is called a *monotone path*, if

$$i_0 < i_1 < i_2 < \ldots < i_k.$$

The *monotone diameter* of G is defined as the smallest integer k, such that for any two indices i and j with $1 \leq i \leq j \leq n$, the vertices x_i and x_j are connected in G by a monotone path that contains at most k edges.

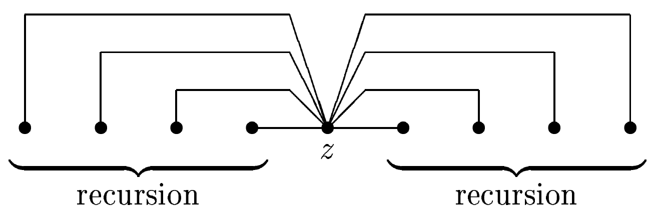
We consider the following shortcutting problem. Given any list L and any positive integer k, construct a graph on the vertices of L having monotone diameter k and that contains as few edges as possible. It should be clear that a solution to this problem can be used to construct t-spanners of low spanner diameter for one-dimensional point sets, even for $t = 1$.

*School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. E-mail: michiel@scs.carleton.ca

1.1 Monotone diameters one, two, and three

The only graph having monotone diameter one is the complete graph, having $\binom{n}{2}$ edges. Algorithm *mono_diam*$(L, n, 2)$, given in Figure 1, constructs a graph having monotone diameter two. (It turns out to be convenient to define this algorithm for any integer $n \geq 0$.)

Monotone diameter two: Consider the list L. We connect each vertex to the middle vertex z. Then we recursively compute a graph having monotone diameter two for those vertices that are less than z. Similarly, we recursively compute a graph having monotone diameter two for those vertices that are larger than z.



Lemma 1.1 *Let L be a list on the vertex set V and let n be the number of vertices of V. The graph $G = (V, E)$ that is computed by algorithm mono_diam$(L, n, 2)$ has monotone diameter less than or equal to two.*

Proof. The proof is by induction on n. If $n \leq 3$, then the claim clearly holds. So let $n \geq 4$ and assume that for any list L' having less than n vertices, algorithm *mono_diam*$(L', |L'|, 2)$ computes a graph on the vertices of L' having monotone diameter less than or equal to two.

Let x and y be any two distinct vertices of L such that x is to the left of y. First assume that x and y are both to the left of the middle vertex z. Consider the edge set E_1 that is computed by our algorithm. The induction hypothesis implies that there is a monotone path, in E_1, between x and y, containing at most two edges. Clearly, this path is also monotone in G. The case when x and y are both to the right of z can be treated in a symmetric way. Assume next that x or y is equal to the middle vertex z. Then x and y are connected by an edge in E. This single edge forms a monotone path in G. The final case is when x is to the left of z, and y is to the right of z. In this case, x and y are connected by the monotone path in G, consisting of the two edges $\{x, z\}$ and $\{z, y\}$. ∎

Algorithm *mono_diam*$(L, n, 2)$
(* L is a list with n vertices. The algorithm returns a graph
  whose monotone diameter is at most two. *)
if $0 \leq n \leq 3$
then $E :=$ edge set of L;
  return E
else $z := \lceil n/2 \rceil$-th vertex of L;
  $L_1 :=$ list containing all vertices of L that are strictly
    to the left of z;
  $E_1 := mono\_diam(L_1, \lceil n/2 \rceil - 1, 2)$;
  $L_2 :=$ list containing all vertices of L that are strictly
    to the right of z;
  $E_2 := mono\_diam(L_2, \lfloor n/2 \rfloor, 2)$;
  $E := E_1 \cup E_2 \cup \{\{x, z\} : x$ is a vertex of L, $x \neq z\}$;
  return E
endif

Figure 1: *Constructing a graph having monotone diameter less than or equal to two.*

Let $F_2(n)$ denote the number of edges in the graph that is computed by algorithm *mono_diam*$(L, n, 2)$. We have

$$F_2(n) = \begin{cases} 0 & \text{if } n = 0, \\ n - 1 & \text{if } 1 \leq n \leq 3, \\ n - 1 + F_2(\lceil n/2 \rceil - 1) + F_2(\lfloor n/2 \rfloor) & \text{if } n \geq 4. \end{cases}$$

Lemma 1.2 *$F_2(n) \leq n \log n$ for all $n \geq 0$.*

Proof. The proof is by induction on n. If $0 \leq n \leq 3$, then the inequality is easy to verify. (Recall that we defined $\log 0 = 0$.) Let $n \geq 4$ and assume that $F_2(k) \leq k \log k$ for all k with $0 \leq k < n$. Then

$$\begin{aligned} F_2(n) &= n - 1 + F_2(\lceil n/2 \rceil - 1) + F_2(\lfloor n/2 \rfloor) \\ &\leq n + (\lceil n/2 \rceil - 1) \log(\lceil n/2 \rceil - 1) + \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor \\ &\leq n + \lceil n/2 \rceil \log(n/2) + \lfloor n/2 \rfloor \log(n/2) \\ &= n + n \log(n/2) \\ &= n \log n. \end{aligned}$$

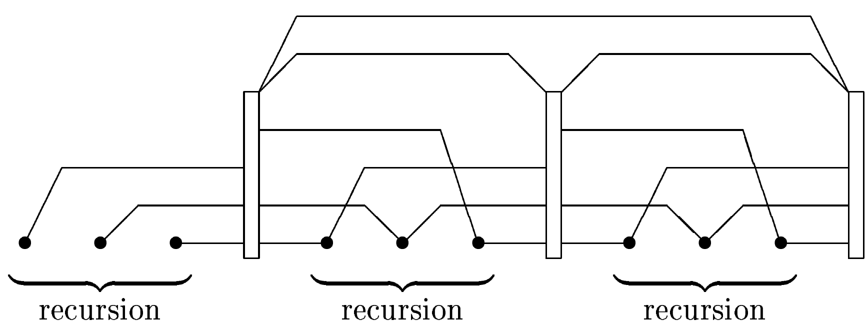
This completes the proof. ∎

By using a similar recurrence, it follows that the running time of algorithm $mono\_diam(L, n, 2)$ is $O(n \log n)$. Hence, we have proved the following result.

Theorem 1.3 *Given a list with n vertices, we can compute in $O(n \log n)$ time a graph on these vertices having at most $n \log n$ edges and whose monotone diameter is less than or equal to two.*

We now turn to the problem of constructing a graph having monotone diameter three. The construction is a generalization of the previous algorithm $mono\_diam(L, n, 2)$.

Monotone diameter three: Consider the list L. Let L' be the list containing every $sqrtn$-th vertex of L. We connect the vertices of L' by a complete graph. Also, we connect each vertex of $L \setminus L'$ to the nearest vertex to its left in L', and to the nearest vertex to its right in L'. The vertices of L' divide the list L into $\sqrt{n}$ sublists, each containing $\sqrt{n} - 1$ vertices. We recursively compute a graph having monotone diameter three for each sublist. At each level of the recursion, $O(n)$ edges are obtained. Since the recursion depth is $O(\log \log n)$, the total number of edges obtained is $O(n \log \log n)$. In the figure below, the dots and rectangles represent the vertices of L.



The formal algorithm, which we denote by $mono\_diam(L, n, 3)$, is given in Figure 2. The following lemma states the correctness of this algorithm. The proof is similar to that of Lemma 1.1.

Lemma 1.4 *Let L be a list on the vertex set V and let n be the number of vertices of V. The graph $G = (V, E)$ that is computed by algorithm $mono\_diam(L, n, 3)$ has monotone diameter less than or equal to three.*

Algorithm *mono_diam*$(L, n, 3)$
(* L is a list with n vertices. The algorithm returns a graph
whose monotone diameter is at most three. *)
if $0 \leq n \leq 4$
then $E :=$ edge set of L;
return E
else number the vertices of L as $x_1, x_2, \ldots, x_n$;
$\ell := \lceil \sqrt{n}\, \rceil$;
$m := \lfloor n/\ell \rfloor$;
$L' :=$ list containing the vertices $x_{i\ell}$, $1 \leq i \leq m$;
$E' :=$ edge set of the complete graph on L';
$E'_1 := \{\{x_{i\ell+j}, x_{(i+1)\ell}\} : 0 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
$E'_2 := \{\{x_{i\ell}, x_{i\ell+j}\} : 1 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
$E'_3 := \{\{x_{m\ell}, x_j\} : m\ell + 1 \leq j \leq n\}$;
for $i := 0$ **to** $m-1$
do $L_i :=$ list containing the vertices x_j, $i\ell + 1 \leq j \leq (i+1)\ell - 1$;
$E_i := \mathit{mono\_diam}(L_i, \ell - 1, 3)$
endfor;
$L_m :=$ list containing the vertices x_j, $m\ell + 1 \leq j \leq n$;
$E_m := \mathit{mono\_diam}(L_m, n - m\ell, 3)$;
$E := E' \cup E'_1 \cup E'_2 \cup E'_3 \cup E_0 \cup E_1 \cup \ldots \cup E_m$;
return E
endif

Figure 2: *Constructing a graph having monotone diameter less than or equal to three.*

Lemma 1.5

We mentioned above that the number of edges in the graph that is computed by algorithm *mono_diam*$(L, n, 3)$ is $O(n \log\log n)$. Let us prove this formally. Let $F_3(n)$ denote the number of edges that are reported by this algorithm, when given a list with n vertices. The function F_3 satisfies the following recurrence.

$$F_3(n) \leq \begin{cases} 0 & \text{if } n = 0, \\ n - 1 & \text{if } 1 \leq n \leq 4, \\ \binom{m}{2} + 2n + m \cdot F_3(\ell - 1) + F_3(n - \ell m) & \text{if } n \geq 5, \end{cases} \qquad (1)$$

where $\ell := \lceil \sqrt{n}\, \rceil$ and $m := \lfloor n/\ell \rfloor$.

$F_3(n) \leq 3n \log\log n + 1$ *for all* $n \geq 0$.

Proof. The proof is by induction on n. For $0 \leq n \leq 4$, the inequality is easy to verify. (Recall that we defined $\log 0 = 0$.) So let $n \geq 5$ and assume that $F_3(k) \leq 3k \log\log k + 1$ for all k with $0 \leq k < n$. First observe that

$$m = \lfloor n/\ell \rfloor \leq n/\ell = n/\lceil \sqrt{n} \rceil \leq \sqrt{n}.$$

It follows that

$$\binom{m}{2} \leq m^2/2 \leq n/2.$$

Since $2 \leq \ell - 1 \leq \sqrt{n} < n$, the induction hypothesis implies that

$$\begin{aligned} F_3(\ell - 1) &\leq 3(\ell - 1)\log\log(\ell - 1) + 1 \\ &\leq 3\ell \log\log \sqrt{n} + 1 \\ &= 3\ell(\log\log n - 1) + 1. \end{aligned}$$

Since $0 \leq n - \ell m < \ell$, we have $n - \ell m \leq \ell - 1 \leq \sqrt{n} < n$. Therefore, the induction hypothesis implies that

$$\begin{aligned} F_3(n - \ell m) &\leq 3(n - \ell m)\log\log(n - \ell m) + 1 \\ &\leq 3(n - \ell m)\log\log \sqrt{n} + 1 \\ &= 3(n - \ell m)(\log\log n - 1) + 1. \end{aligned}$$

Combining these inequalities with the recurrence (1), it follows that

$$\begin{aligned} F_3(n) &\leq 5n/2 + m(3\ell(\log\log n - 1) + 1) + 3(n - \ell m)(\log\log n - 1) + 1 \\ &= 3n \log\log n + 1 + m - n/2 \\ &\leq 3n \log\log n + 1, \end{aligned}$$

where the last inequality follows from the fact that $m \leq n/2$. ∎

A similar recurrence as the one for F_3 shows that the running time of algorithm *mono_diam*$(L, n, 3)$ is $O(n \log\log n)$. We summarize our result.

Theorem 1.6 *Given a list with* n *vertices, we can compute in* $O(n \log\log n)$ *time a graph on these vertices having at most* $3n \log\log n + 1$ *edges and whose monotone diameter is less than or equal to three.*

1.2 Generalization to higher monotone diameters

Our goal is to generalize the results of Theorems 1.3 and 1.6 to monotone diameters k that are greater than three. We first describe the idea for $k = 4$. Assume that n is a power of two. Let $\ell := \log n$ and consider the list $L = (x_1, x_2, \ldots, x_n)$. Let L' be the list containing the vertices $x_{i\ell}$, $1 \leq i \leq n/\ell$. We connect the vertices of L' by a graph having monotone diameter two. By Theorem 1.3, there is such a graph having $O(n)$ edges. Next, we connect each vertex of $L \setminus L'$ to the nearest vertex to its left in L', and to the nearest vertex to its right in L'. This also gives $O(n)$ edges. The vertices of L' divide the list L into n/ℓ sublists, each containing $\ell - 1$ vertices. We recursively compute a graph having monotone diameter four, for each of these sublists.

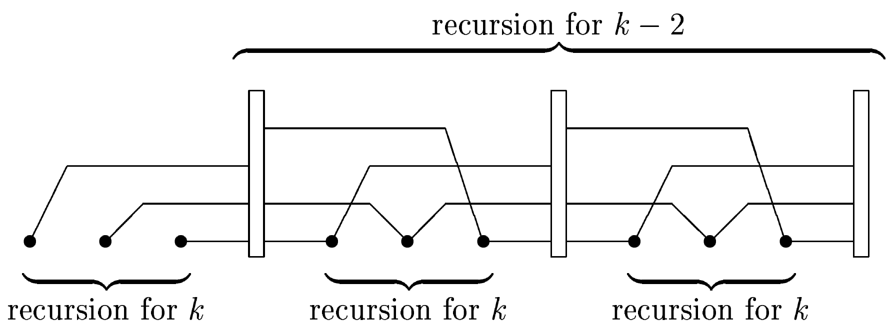
As in algorithm $\mathit{mono\_diam}(L, n, 3)$, $O(n)$ edges are added at each level of the recursion. The recursion depth is bounded from above by $\log^* n$, which is defined as

$$\log^* n := \min\{s \geq 0 : \underbrace{\log \log \ldots \log}_{s} n \leq 1\}.$$

(Observe that $\log^* 0 = \log^* 1 = 0$.) Hence, the entire graph will contain $O(n \log^* n)$ edges.

This idea can be generalized in the following way to an arbitrary monotone diameter k.

Monotone diameter k: Consider the list L. We choose an integer ℓ_k and construct a list L' containing every ℓ_k-th element of L. Then we compute a graph on the vertices of L' having monotone diameter $k-2$. Moreover, we connect each vertex of $L \setminus L'$ to its left and right neighbors in L'. The vertices of L' divide L into sublists. For each sublist, we recursively compute a graph having monotone diameter k. We choose ℓ_k such that the number of edges that are added at each level of the recursion is $O(n)$. In this way, the total number of edges in the final graph is "small". It turns out that the "correct" value of ℓ_k is related to the functional inverse of the Ackermann function. In the figure below, the dots and rectangles represent the vertices of L.



1.3 The Ackermann function and its inverse

The set of non-negative integers will be denoted by $\mathbb{N}$. We will use the following notation. For any function $f : \mathbb{N} \longrightarrow \mathbb{N}$ and any $s \in \mathbb{N}$, we denote the s-fold iteration of f by $f^{(s)}$. That is, the functions $f^{(s)} : \mathbb{N} \longrightarrow \mathbb{N}$ are inductively defined by

$$f^{(0)}(n) := n \text{ for all } n \geq 0,$$

and

$$f^{(s)}(n) := f(f^{(s-1)}(n)) \text{ for all } n \geq 0 \text{ and } s \geq 1.$$

Definition 1.7 For each $k \geq 0$, the functions $A_k : \mathbb{N} \longrightarrow \mathbb{N}$ and $B_k : \mathbb{N} \longrightarrow \mathbb{N}$ are recursively defined as follows:

$$A_0(n) \;\; := \;\; 2n \text{ for all } n \geq 0,$$

$$
\begin{aligned}
A_k(n) &:= \begin{cases} 1 & \text{if } k \geq 1 \text{ and } n = 0, \\ A_{k-1}(A_k(n-1)) & \text{if } k \geq 1 \text{ and } n \geq 1, \end{cases} \\
B_0(n) &:= n^2 \text{ for all } n \geq 0, \\
B_k(n) &:= \begin{cases} 2 & \text{if } k \geq 1 \text{ and } n = 0, \\ B_{k-1}(B_k(n-1)) & \text{if } k \geq 1 \text{ and } n \geq 1. \end{cases}
\end{aligned}
$$

The following lemma gives an alternative way for computing the functions A_k and B_k. The claims can be proved by a straightforward induction on n.

Lemma 1.8 *For all $k \geq 1$ and $n \geq 0$, we have*

1. $A_k(n) = A_{k-1}^{(n)}(1)$ *and*
2. $B_k(n) = B_{k-1}^{(n)}(2)$.

Let us consider some examples. Using Definition 1.7 or Lemma 1.8, it can easily be verified that for all $n \geq 0$,

- $A_1(n) = 2^n$,
- $A_2(n) = \underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}$,
- $B_1(n) = 2^{2^n}$,
- $B_2(n) = \underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{2n+1}$.

It should be clear from these examples that, for $k \geq 2$, the functions A_k and B_k are extremely fast growing. The next four lemmas state some useful monotonicity properties of the functions A_k and B_k.

Lemma 1.9 *For all $k \geq 0$ and $n \geq 0$, we have*

1. $A_k(n) \geq 2n$ *and*
2. $B_k(n) \geq n^2$.

Proof. The claims can be proved by double inductions on k and n. ∎

Lemma 1.10 *For all $k \geq 0$, the functions A_k and B_k are non-decreasing.*

Proof. It follows immediately from the definition of A_0 that this function is non-decreasing. Let $k \geq 1$. We will show that $A_k(n) \geq A_k(n-1)$ for all $n \geq 1$. By Lemma 1.9, we have

$$A_{k-1}(A_k(n-1)) \geq 2 \cdot A_k(n-1) \geq A_k(n-1).$$

Therefore,

$$A_k(n) = A_{k-1}(A_k(n-1)) \geq A_k(n-1).$$

The proof that the functions B_k are non-decreasing is similar. ∎

Using Lemmas 1.9 and 1.10, the following lemma can easily be proved.

Lemma 1.11 *For all $k \geq 0$ and $n \geq 0$, we have $A_{k+1}(n) \geq A_k(n)$.*

Lemma 1.12 *For all $k \geq 0$ and $n \geq 3$, we have $A_k(n+1) \leq A_{k+1}(n)$.*

Proof. First observe that, by Lemma 1.11,

$$A_{k+1}(n-1) \geq A_0(n-1) = 2n-2 \geq n+1.$$

Since the function A_k is non-decreasing, it follows that

$$A_{k+1}(n) = A_k(A_{k+1}(n-1)) \geq A_k(n+1),$$

which is what we wanted to show. ∎

We now define the functional inverses of the functions A_k and B_k.

Definition 1.13 For each $k \geq 0$, we define the functions $\alpha_{2k} : \mathbb{N} \longrightarrow \mathbb{N}$ and $\alpha_{2k+1} : \mathbb{N} \longrightarrow \mathbb{N}$ by

1. $\alpha_{2k}(n) := \min\{s \geq 0 : A_k(s) \geq n\}$ for all $n \geq 0$, and
2. $\alpha_{2k+1}(n) := \min\{s \geq 0 : B_k(s) \geq n\}$ for all $n \geq 0$.

Observe that by Lemma 1.9, these functions are well-defined. Let us look at some examples. For all $n \geq 0$, we have

- $\alpha_0(n) = \lceil n/2 \rceil$,
- $\alpha_1(n) = \lceil \sqrt{n}\, \rceil$,

- $\alpha_2(n) = \lceil \log n \rceil$,
- $\alpha_3(n) = \lceil \log \log n \rceil$,
- $\alpha_4(n) = \log^* n$,
- $\alpha_5(n) = \left\lfloor \frac{1}{2} \log^* n \right\rfloor$.

Lemma 1.14 *For each $k \geq 0$, the function α_k is non-decreasing.*

Proof. We will prove the claim for even values of k. (For odd values of k, the proof is similar.) For simplicity, we write $2k$ instead of k. Let m and n be two non-negative integers such that $m < n$. We will prove that $\alpha_{2k}(m) \leq \alpha_{2k}(n)$.

Let $s := \alpha_{2k}(n)$. By the definition of the function α_{2k}, we have $A_k(s) \geq n$. Since $m < n$, we also have $A_k(s) \geq m$. Then the definition of α_{2k} implies that $\alpha_{2k}(m) \leq s$, i.e., $\alpha_{2k}(m) \leq \alpha_{2k}(n)$. ∎

In Lemma 1.17 below, we will state a useful characterization of the functions α_k. Before we can prove it, we need two lemmas.

Lemma 1.15 *For each $k \geq 1$, we have*

1. *$\alpha_{2k}(n) = 1 + \alpha_{2k}(\alpha_{2k-2}(n))$ for all $n \geq 2$, and*
2. *$\alpha_{2k+1}(n) = 1 + \alpha_{2k+1}(\alpha_{2k-1}(n))$ for all $n \geq 3$.*

Proof. Let $k \geq 1$ and $n \geq 2$. Since the function A_{k-1} is non-decreasing, it follows from the definition of the function α_{2k-2} that for all $m \geq 0$,

$$A_{k-1}(m) \geq n \text{ if and only if } m \geq \alpha_{2k-2}(n).$$

By using this equivalence, we get the following chain of equalities:

$$\begin{aligned}
\alpha_{2k}(n) &= \min\{s \geq 0 : A_k(s) \geq n\} \\
&= \min\{s \geq 1 : A_k(s) \geq n\} \\
&= \min\{s \geq 1 : A_{k-1}(A_k(s-1)) \geq n\} \\
&= \min\{s \geq 1 : A_k(s-1) \geq \alpha_{2k-2}(n)\} \\
&= 1 + \min\{s' \geq 0 : A_k(s') \geq \alpha_{2k-2}(n)\} \\
&= 1 + \alpha_{2k}(\alpha_{2k-2}(n)).
\end{aligned}$$

The second claim can be proved in a similar way. ∎

Lemma 1.16 *Let $k \geq 0$.*

1. *For each $n \geq 2$, there is an $s \geq 1$ such that $\alpha_{2k}^{(s)}(n) \leq 1$.*
2. *For each $n \geq 3$, there is an $s \geq 1$ such that $\alpha_{2k+1}^{(s)}(n) \leq 2$.*

Proof. We will prove the first claim, and leave the proof of the second claim to the reader. By Lemma 1.9, we have $A_k(m-1) \geq 2(m-1) \geq m$ for all $m \geq 2$. Then the definition of the function α_{2k} implies that

$$\alpha_{2k}(m) \leq m-1 \text{ for all } m \geq 2. \tag{2}$$

Now assume that $\alpha_{2k}^{(s)}(n) \geq 2$ for all $s \geq 1$. For $s = n$, this reads $\alpha_{2k}^{(n)}(n) \geq 2$. On the other hand, by repeatedly applying (2), we obtain

$$\begin{aligned}
\alpha_{2k}^{(n)}(n) &= \alpha_{2k}(\alpha_{2k}^{(n-1)}(n)) \\
&\leq \alpha_{2k}^{(n-1)}(n) - 1 \\
&\leq \alpha_{2k}^{(n-2)}(n) - 2 \\
&\vdots \\
&\leq \alpha_{2k}^{(1)}(n) - (n-1) \\
&= \alpha_{2k}(n) - (n-1) \\
&\leq 0,
\end{aligned}$$

which is a contradiction. ∎

Lemma 1.17 *For all $k \geq 1$ and $n \geq 0$, we have*

1. $\alpha_{2k}(n) = \min\{s \geq 0 : \alpha_{2k-2}^{(s)}(n) \leq 1\}$, *and*
2. $\alpha_{2k+1}(n) = \min\{s \geq 0 : \alpha_{2k-1}^{(s)}(n) \leq 2\}$.

Proof. We only prove the first claim. The second claim can be proved in a similar way. If $n \in \{0,1\}$, then the first claim follows from the fact that $\alpha_{2k}(n) = 0$. So let $n \geq 2$. Let $s \geq 1$ be the smallest integer such that $\alpha_{2k-2}^{(s)}(n) \leq 1$. By Lemma 1.16, s is well-defined. Observe that $\alpha_{2k-2}^{(j)}(n) \geq 2$ for all j with $0 \leq j < s$. By applying Lemma 1.15 twice, we get

$$\begin{aligned}
\alpha_{2k}(n) &= 1 + \alpha_{2k}(\alpha_{2k-2}(n)) \\
&= 2 + \alpha_{2k}(\alpha_{2k-2}(\alpha_{2k-2}(n))) \\
&= 2 + \alpha_{2k}(\alpha_{2k-2}^{(2)}(n)).
\end{aligned}$$

Repeating this, we get

$$
\begin{aligned}
\alpha_{2k}(n) &= 3 + \alpha_{2k}(\alpha_{2k-2}^{(3)}(n)) \\
&= 4 + \alpha_{2k}(\alpha_{2k-2}^{(4)}(n)) \\
&\vdots \\
&= (s-1) + \alpha_{2k}(\alpha_{2k-2}^{(s-1)}(n)) \\
&= s + \alpha_{2k}(\alpha_{2k-2}^{(s)}(n)).
\end{aligned}
$$

Since $\alpha_{2k-2}^{(s)}(n) \in \{0,1\}$, we have $\alpha_{2k}(\alpha_{2k-2}^{(s)}(n)) = 0$. Hence, $\alpha_{2k}(n) = s$, which is exactly what we wanted to show. ∎

We now define the Ackermann function A and its functional inverse α.

Definition 1.18 (Ackermann function) The *Ackermann function* $A : \mathbb{N} \longrightarrow \mathbb{N}$ is defined by

$$A(n) := A_n(n) \text{ for all } n \geq 0.$$

The reader can easily verify that $A(0) = 0$, $A(1) = 2$, $A(2) = 4$, $A(3) = 2^{16} = 65{,}536$. Moreover, we have

$$A(4) = A_3 \left(\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{65{,}536} \right).$$

Definition 1.19 (inverse Ackermann function) The *inverse Ackermann function* $\alpha : \mathbb{N} \longrightarrow \mathbb{N}$ is defined by

$$\alpha(n) := \min\{s \geq 0 : A(s) \geq n\} \text{ for all } n \geq 0.$$

By Lemma 1.9, we have $A(n) = A_n(n) \geq 2n \geq n$, for all $n \geq 0$. Therefore, the function α is well-defined. It is not difficult to verify that $\alpha(0) = 0$, $\alpha(1) = 1$, $\alpha(2) = 1$, $\alpha(3) = 2$, and $\alpha(65{,}536) = 3$. Although the function α is unbounded, it grows extremely slowly. In fact, for all practical applications, we have $\alpha(n) \leq 4$.

Lemma 1.20 *The function α is non-decreasing.*

Proof. The proof is similar to that of Lemma 1.14. ∎

We now consider the behavior of the function $\alpha_{2k}(n)$ for values of k that are close to $\alpha(n)$. Observe that for such k, the index of the function α_{2k} depends on n.

Lemma 1.21 *The following inequalities hold:*

1. $\alpha_{2\alpha(n)-2}(n) \geq \alpha(n)$ *for all* $n \geq 1$,
2. $\alpha_{2\alpha(n)}(n) \leq \alpha(n)$ *for all* $n \geq 0$, *and*
3. $\alpha_{2\alpha(n)+2}(n) \leq 4$ *for all* $n \geq 0$.

Proof. Let $n \geq 1$. The definition of the function α implies that

$$A_{\alpha(n)-1}(\alpha(n)-1) = A(\alpha(n)-1) < n.$$

Combining this with the definition of the function $\alpha_{2\alpha(n)-2}(n)$, i.e.,

$$\alpha_{2\alpha(n)-2}(n) = \min\{s \geq 0 : A_{\alpha(n)-1}(s) \geq n\},$$

and the fact that the function $A_{\alpha(n)-1}$ is non-decreasing, we obtain $\alpha_{2\alpha(n)-2}(n) > \alpha(n) - 1$. Since $\alpha_{2\alpha(n)-2}(n)$ and $\alpha(n)$ are integers, this proves the first inequality.

To prove the second inequality, let $n \geq 0$. The definition of the function α implies that

$$A_{\alpha(n)}(\alpha(n)) = A(\alpha(n)) \geq n.$$

Since

$$\alpha_{2\alpha(n)}(n) = \min\{s \geq 0 : A_{\alpha(n)}(s) \geq n\},$$

it follows that $\alpha_{2\alpha(n)}(n) \leq \alpha(n)$.

It remains to prove the third inequality. Let $n \geq 0$. Lemma 1.12 implies that for all $k \geq 0$,

$$A_{k+1}(3) \geq A_k(4) \geq A_{k-1}(5) \geq \ldots \geq A_0(k+4) = 2(k+4) \geq k.$$

Combining this inequality with the fact that the function A_k is non-decreasing, we get

$$A_{k+1}(4) = A_k(A_{k+1}(3)) \geq A_k(k) = A(k),$$

for all $k \geq 0$. For $k := \alpha(n)$, this reads

$$A_{\alpha(n)+1}(4) \geq A(\alpha(n)).$$

By the definition of α, we have $A(\alpha(n)) \geq n$. Hence,

$$A_{\alpha(n)+1}(4) \geq n.$$

Then the definition of the function value $\alpha_{2\alpha(n)+2}(n)$, i.e.,

$$\alpha_{2\alpha(n)+2}(n) = \min\{s \geq 0 : A_{\alpha(n)+1}(s) \geq n\},$$

immediately implies that $\alpha_{2\alpha(n)+2}(n) \leq 4$. ∎

1.4 Computing the α-values

See La Poutre's thesis.

1.5 Monotone diameters larger than three

We are now ready to solve the shortcutting problem for a list L with n vertices and monotone diameter k, for values of k that are greater than or equal to four. The basic idea was given in Section 1.2. The formal algorithm, which we denote by $mono\_diam(L, n, k)$, is given in Figure 3.

Lemma 1.22 *Let L be a list on the vertex set V, let n be the number of vertices of V, and let $k \geq 4$. The graph $G = (V, E)$ that is computed by algorithm $mono\_diam(L, n, k)$ has monotone diameter less than or equal to k.*

Proof. We prove that the algorithm terminates. Using this, the claim about the monotone diameter can be proved by a straightforward induction on k.

Let $k \geq 4$ and $n \geq k+2$. Consider the non-negative integers $\ell := \alpha_{k-2}(n)$ and $m := \lfloor n/\ell \rfloor$ that are used in the algorithm. Since $n \geq 6$, it follows easily from the definition of the function α_{k-2} that $\ell \geq 1$. Also, by (2), we have $\ell = \alpha_{k-2}(n) \leq n - 1$ if $k - 2$ is even. It is easy to verify that this inequality also holds if $k - 2$ is odd. Hence, we know that $0 \leq \ell - 1 \leq n - 2 < n$. This shows that the list L_i in the recursive call $mono\_diam(L_i, \ell - 1, k)$ has less than n vertices. It is also easy to see that $0 \leq n - m\ell < \ell < n$, which implies

Algorithm *mono_diam*(L, n, k)
($*$ L is a list with n vertices and $k \geq 4$. The algorithm returns a graph whose monotone diameter is at most k. $*$)
if $0 \leq n \leq k+1$
then $E :=$ edge set of L;
    return E
else number the vertices of L as $x_1, x_2, \ldots, x_n$;
    $\ell := \alpha_{k-2}(n)$;
    $m := \lfloor n/\ell \rfloor$;
    $L' :=$ list containing the vertices $x_{i\ell}$, $1 \leq i \leq m$;
    $E' := mono\_diam(L', m, k-2)$;
    $E'_1 := \{\{x_{i\ell+j}, x_{(i+1)\ell}\} : 0 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
    $E'_2 := \{\{x_{i\ell}, x_{i\ell+j}\} : 1 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
    $E'_3 := \{\{x_{m\ell}, x_j\} : m\ell + 1 \leq j \leq n\}$;
    for $i := 0$ **to** $m-1$
    do $L_i :=$ list containing the vertices x_j, $i\ell + 1 \leq j \leq (i+1)\ell - 1$;
        $E_i := mono\_diam(L_i, \ell - 1, k)$
    endfor;
    $L_m :=$ list containing the vertices x_j, $m\ell + 1 \leq j \leq n$;
    $E_m := mono\_diam(L_m, n - m\ell, k)$;
    $E := E' \cup E'_1 \cup E'_2 \cup E'_3 \cup E_0 \cup E_1 \cup \ldots \cup E_m$;
    return E
endif

Figure 3: *Constructing a graph having monotone diameter less than or equal to k.*

---

that the list L_m in the recursive call *mono_diam*$(L_m, n - m\ell, k)$ has less than n vertices. ∎

For each $k \geq 2$, we denote by $F_k(n)$ the number of edges in the graph that is computed by algorithm *mono_diam*(L, n, k), when given a list L with n vertices. Lemmas 1.2 and 1.5 give upper bounds for the functions F_2 and F_3. From algorithm *mono_diam*(L, n, k), we obtain the following recurrence for the functions F_k with $k \geq 4$:

$$F_k(n) \le \begin{cases} 0 & \text{if } n = 0, \\ n-1 & \text{if } 1 \le n \le k+1, \\ 2n + F_{k-2}(m) + m \cdot F_k(\ell - 1) + F_k(n - \ell m) & \text{if } n \ge k+2, \end{cases} \tag{3}$$

where $\ell := \alpha_{k-2}(n)$ and $m := \lfloor n/\ell \rfloor$.

Lemma 1.23 *For all $k \ge 2$ and $n \ge 0$, we have*

$$F_k(n) \le kn\,\alpha_k(n) + n.$$

Proof. The proof is by a double induction on k and n. For $k = 2$, the claim follows immediately from Lemma 1.2. If $k = 3$ and $n = 0$, then the claim follows from the fact that $F_3(0) = 0$. For $k = 3$ and $n \ge 1$, the claim follows from Lemma 1.5. Let $k \ge 4$ and assume that

$$F_{k-2}(s) \le (k-2)s\,\alpha_{k-2}(s) + s \tag{4}$$

for all $s \ge 0$. We will prove that

$$F_k(n) \le kn\,\alpha_k(n) + n \tag{5}$$

for all $n \ge 0$. If $n = 0$, then (5) holds, because $F_k(0) = 0$. If $1 \le n \le k+1$, then (5) follows from the fact that $F_k(n) \le n - 1$. So let $n \ge k+2$ and assume that

$$F_k(s) \le ks\,\alpha_k(s) + s \tag{6}$$

for all s with $0 \le s < n$. Let $\ell := \alpha_{k-2}(n)$ and $m := \lfloor n/\ell \rfloor$. Then $m \le n/\alpha_{k-2}(n) \le n$. Since the function α_{k-2} is non-decreasing, it follows that

$$m\,\alpha_{k-2}(m) \le m\,\alpha_{k-2}(n) = m\ell \le n.$$

Hence, by the induction hypothesis (4), we obtain

$$F_{k-2}(m) \le (k-2)m\,\alpha_{k-2}(m) + m \le (k-2)n + m.$$

We saw in the proof of Lemma 1.22 that $0 \le \ell - 1 < n$. Therefore, by the induction hypothesis (6), we have

$$\begin{aligned} F_k(\ell - 1) &\le k(\ell - 1)\,\alpha_k(\ell - 1) + \ell - 1 \\ &\le k(\ell - 1)\,\alpha_k(\ell) + \ell - 1. \end{aligned}$$

Since $0 \leq n - \ell m < \ell < n$, it follows from the induction hypothesis (6) that

$$\begin{aligned} F_k(n - \ell m) &\leq k(n - \ell m)\,\alpha_k(n - \ell m) + n - \ell m \\ &\leq k(n - \ell m)\,\alpha_k(\ell) + n - \ell m. \end{aligned}$$

Combining these inequalities with the recurrence (3), we get

$$\begin{aligned} F_k(n) &\leq 2n + (k-2)n + m + mk(\ell - 1)\,\alpha_k(\ell) + m(\ell - 1) \\ &\qquad + k(n - \ell m)\,\alpha_k(\ell) + n - \ell m \\ &= k(n - m)\,\alpha_k(\ell) + (k+1)n. \end{aligned}$$

By Lemma 1.15, we have

$$\alpha_k(n) = 1 + \alpha_k(\alpha_{k-2}(n)) = 1 + \alpha_k(\ell).$$

We conclude that

$$\begin{aligned} F_k(n) &\leq k(n - m)\,(\alpha_k(n) - 1) + (k+1)n \\ &= kn\,\alpha_k(n) + n + km(1 - \alpha_k(n)) \\ &\leq kn\,\alpha_k(n) + n, \end{aligned}$$

where the last inequality follows from the fact that $\alpha_k(n) \geq 1$. ■

Let $T_k(n)$ denote the running time of algorithm $mono\_diam(L, n, k)$, when given a list L with n vertices. There is a positive constant c such that

$$T_k(n) \leq \begin{cases} c & \text{if } 0 \leq n \leq k+1, \\ cn + T_{k-2}(m) + m \cdot T_k(\ell - 1) + T_k(n - \ell m) & \text{if } n \geq k+2, \end{cases}$$

where $\ell := \alpha_{k-2}(n)$ and $m := \lfloor n/\ell \rfloor$. This recurrence solves to $T_k(n) = O(kn\,\alpha_k(n))$, where the constant factor in the Big-Oh bound is independent of k. We have proved the following result.

Theorem 1.24 *Given a list with n vertices and an integer $k \geq 2$, we can compute, in $O(kn\,\alpha_k(n))$ time, a graph on these vertices having at most $kn\,\alpha_k(n) + n$ edges and whose monotone diameter is less than or equal to k. The constant in the Big-Oh bound does not depend on k.*

1.6 Shortcutting a list using $O(n)$ edges

In this section, we consider shortcuttings of a list with n vertices that use $O(n)$ edges. Observe that Theorem 1.24 does not give such a shortcutting.

> **Bucketing:** Let $L = (x_1, x_2, \ldots, x_n)$ be a list, $\ell := 2\alpha(n) + 6$, and $m := n/\ell$. Let L' be the list containing every ℓ-th vertex of L. We apply Theorem 1.24, with $k = 2\alpha(n) + 2$, to L'. Hence, we connect the vertices of L' by a graph whose monotone diameter is at most $2\alpha(n)+2$ and whose number of edges is $O(km\,\alpha_k(m)) = O(km\,\alpha_k(n))$. Since $\alpha_k(n) \leq 4$—see Lemma 1.21—the graph on L' has $O(n)$ edges. The vertices of L' divide L into m sublists, each containing $\ell - 1$ vertices. We connect each sublist using the edges of L; hence no new edges are added to the sublist. Finally, we connect each vertex of $L \setminus L'$ to its left and right neighbors in L'.

The formal algorithm, which we denote by *lin_mono_diam*(L, n), is given in Figure 4.

Theorem 1.25 *Algorithm lin_mono_diam(L, n) computes a graph on the n vertices of the list L having at most $7n$ edges and whose monotone diameter is less than or equal to $2\alpha(n) + 4$. The running time of this algorithm is $O(n)$.*

Proof. Let V be the vertex set of L. Consider the edge set E that is returned by the algorithm and let $G = (V, E)$ be the corresponding graph.

If $0 \leq n \leq 2\alpha(n) + 5$, then the theorem clearly holds. Assume that $n \geq 2\alpha(n) + 6$. Let i and j be two indices such that $1 \leq i < j \leq n$, and consider the vertices x_i and x_j of the list L. We will prove that x_i and x_j are connected, in G, by a monotone path having at most $2\alpha(n) + 4$ edges.

If both x_i and x_j are contained in L', then there is a monotone path in G between them, having at most $k = 2\alpha(n) + 2$ edges. If one of x_i and x_j is contained in L', then G contains a monotone path between them having at most $k + 1 = 2\alpha(n) + 3$ edges. The list L' divides the list L into sublists of length less than or equal to $\ell - 1$, where $\ell = 2\alpha(n) + 6$. If x_i and x_j are in different sublists, then G contains a monotone path between them having at most $k + 2 = 2\alpha(n) + 4$ edges. The remaining case is when x_i and x_j are contained in the same sublist. Since a sublist contains at most $\ell - 1$ vertices,

Algorithm *lin_mono_diam*(L, n)
(* L is a list with n vertices. The algorithm returns a graph with $O(n)$ edges whose monotone diameter is at most $2\alpha(n)+4$. *)
$k := 2\alpha(n) + 2$;
if $0 \leq n \leq k+3$
then $E :=$ edge set of L;
    return E
else number the vertices of L as $x_1, x_2, \ldots, x_n$;
    $\ell := 2\alpha(n) + 6$;
    $m := \lfloor n/\ell \rfloor$;
    $L' :=$ list containing the vertices $x_{i\ell}$, $1 \leq i \leq m$;
    $E' := mono\_diam(L', m, k)$;
    $E'_1 := \{\{x_{i\ell+j}, x_{(i+1)\ell}\} : 0 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
    $E'_2 := \{\{x_{i\ell}, x_{i\ell+j}\} : 1 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
    $E'_3 := \{\{x_{m\ell}, x_j\} : m\ell + 1 \leq j \leq n\}$;
    for $i := 0$ **to** $m-1$
    do $E_i := \{\{x_j, x_{j+1} : i\ell + 1 \leq j \leq (i+1)\ell - 2\}$
    endfor;
    $E_m := \{\{x_j, x_{j+1} : \ell m + 1 \leq j \leq n-1\}$;
    $E := E' \cup E'_1 \cup E'_2 \cup E'_3 \cup E_0 \cup E_1 \cup \ldots \cup E_m$;
    return E
endif

Figure 4: *Shortcutting a list using $O(n)$ edges.*

---

it follows that in this case, there is a monotone path in G between x_i and x_j having at most $\ell - 2 = 2\alpha(n) + 4$ edges. This proves that the monotone diameter of the graph G is less than or equal to $2\alpha(n) + 4$.

Next, we prove the upper bound on the number of edges of the graph G. Consider the edge set E' that is computed by the algorithm. By Theorem 1.24, we have

$$|E'| \leq km\,\alpha_k(m) + m.$$

We know from Lemma 1.21 that $\alpha_k(n) \leq 4$. It follows that

$$\begin{aligned} |E'| &\leq k(n/\ell)\,\alpha_k(n) + n/\ell \\ &\leq (4k+1)n/\ell \end{aligned}$$

$$
\begin{aligned}
&= \frac{8\alpha(n)+9}{2\alpha(n)+6}n \\
&\leq 4n.
\end{aligned}
$$

Finally, it is easy to see that the total size of the other edge sets that are computed by the algorithm, i.e., $E'_1, E'_2, E'_3, E_0, E_1, \ldots, E_m$, is bounded from above by $3n$. This proves that the graph G contains at most $7n$ edges.

The $O(n)$-bound on the running time of algorithm *lin_mono_diam*(L, n) follows in a similar way. ■

We now show how the monotone diameter can be reduced to $2\alpha(n) + 2$, while still using $O(n)$ edges. In fact, this improved solution even uses less edges than the solution of Theorem 1.25.

> **Bucketing and skip lists:** Consider the integers k and ℓ, and the list L' in algorithm *lin_mono_diam*(L, n). This list L' divides L into sublists, each containing $\ell - 1$ vertices. We connected the vertices within each sublist by a list, having monotone diameter $\ell - 2$. In our improved solution, we connect the vertices within a sublist by a deterministic skip list. In this way, the monotone diameter for each sublist will be at most $2\log(\ell - 1)$. As we will see, by choosing the integers k and ℓ slightly smaller and larger, respectively, than in algorithm *lin_mono_diam*(L, n), we improve upon the result of Theorem 1.25.

In the rest of this section, we will formalize this idea.

Lemma 1.26 *Let L be a list with n vertices. In $O(n)$ time, we can compute a graph on these vertices having at most $2n$ edges and whose monotone diameter is less than or equal to $2\log n$.*

Proof. The graph is a deterministic skip list on the vertices of L. ■

We denote the algorithm that takes as input a list L with n vertices and returns the graph of Lemma 1.26, by *skip_list*(L, n). The improved algorithm, which we denote by *lin_mono_diam*$'(L, n)$, is given in Figure 5.

Theorem 1.27 *Algorithm lin_mono_diam*$'(L, n)$ *computes a graph on the n vertices of the list L having at most $4n + o(n)$ edges and whose monotone*

Algorithm $lin\_mono\_diam'(L, n)$
(* L is a list with n vertices. The algorithm returns a graph with $O(n)$ edges whose monotone diameter is at most $2\alpha(n) + 2$. *)
$k := 2\alpha(n)$;
if $0 \leq n \leq k + 3$
then $E :=$ edge set of L;
    return E
else number the vertices of L as $x_1, x_2, \ldots, x_n$;
  $\ell := 1 + 2^{1+\alpha(n)}$;
  $m := \lfloor n/\ell \rfloor$;
  $L' :=$ list containing the vertices $x_{i\ell}$, $1 \leq i \leq m$;
  $E' := mono\_diam(L', m, k)$;
  $E'_1 := \{\{x_{i\ell+j}, x_{(i+1)\ell}\} : 0 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
  $E'_2 := \{\{x_{i\ell}, x_{i\ell+j}\} : 1 \leq i \leq m-1, 1 \leq j \leq \ell - 1\}$;
  $E'_3 := \{\{x_{m\ell}, x_j\} : m\ell + 1 \leq j \leq n\}$;
  for $i := 0$ **to** $m - 1$
  do $L_i :=$ list containing the vertices x_j, $i\ell + 1 \leq j \leq (i+1)\ell - 1$;
    $E_i := skip\_list(L_i, \ell - 1)$
  endfor;
  $L_m :=$ list containing the vertices x_j, $m\ell + 1 \leq j \leq n$;
  $E_m := skip\_list(L_m, n - m\ell)$;
  $E := E' \cup E'_1 \cup E'_2 \cup E'_3 \cup E_0 \cup E_1 \cup \ldots \cup E_m$;
  return E
endif

Figure 5: *The improved shortcutting algorithm.*

---

diameter is less than or equal to $2\alpha(n)+2$. *The running time of this algorithm is* $O(n)$.

Proof. Let V be the vertex set of L. Consider the edge set E that is returned by the algorithm and let $G = (V, E)$ be the corresponding graph. If $0 \leq n \leq 2\alpha(n) + 3$, then the theorem clearly holds. Assume that $n \geq 2\alpha(n) + 4$. By considering different cases as in the proof of Theorem 1.25, and using Lemma 1.26, it follows that the monotone diameter of the graph G is bounded from above by

$$\max(k + 2, 2\log(\ell - 1)) = 2\alpha(n) + 2.$$

Next, we analyze the number of edges of the graph G. First consider the edge set E' that is computed by the algorithm. It follows from Theorem 1.24 that

$$|E'| \leq km\,\alpha_k(m) + m = (k\,\alpha_k(m) + 1)m.$$

By Lemma 1.21, we have $\alpha_k(n) = \alpha_{2\alpha(n)}(n) \leq \alpha(n)$. Therefore,

$$\begin{aligned} |E'| &\leq (k\,\alpha_k(n) + 1)m \\ &\leq (k\,\alpha(n) + 1)n/\ell \\ &= \frac{2(\alpha(n))^2 + 1}{1 + 2^{1+\alpha(n)}}\,n \\ &= o(n). \end{aligned}$$

By Lemma 1.26, the total size of the edge sets $E_0, E_1, \ldots, E_m$ is less than or equal to $2n$. Finally, the lists E'_1, E'_2, and E'_3 together have size at most $2n$.

This proves that the graph G contains at most $4n+o(n)$ edges. In a similar way, it follows that the running time of algorithm *lin_mono_diam*$'(L, n)$ is $O(n)$. ∎