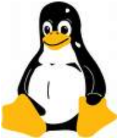
Cyclic Increasing Sequences

Michael Albert[1] Mike Atkinson[1] Doron Nussbaum[2] Jörg Sack[2] Nicola Santoro[2]

[1]Department of Computer Science, University of Otago

[2]School of Computer Science, Carleton University

St Andrews, 5 October, 2006



Outline of talk

Increasing subsequences

- A sequence of numbers: 4 3 1 6 2 5 7

Increasing subsequences

- A sequence of numbers: 4 3 1 6 2 5 7
- An increasing subsequence: 4 3 1 6 2 5 7

Increasing subsequences

- A sequence of numbers: 4 3 1 6 2 5 7
- An increasing subsequence: 4 3 1 6 2 5 7
- A longer increasing subsequence: 4 3 1 6 2 5 7

Increasing subsequences

- A sequence of numbers: 4 3 1 6 2 5 7
- An increasing subsequence: 4 3 1 6 2 5 7
- A longer increasing subsequence: 4 3 1 6 2 5 7
- The longest increasing subsequence: 4 3 1 6 2 5 7

How Long?

- $1\ 2\ 3\ \cdots\ n$: LIS has length n

How Long?

- $1\ 2\ 3\ \cdots\ n$: LIS has length n
- $n\ n-1\ \cdots 2\ 1$: LIS has length 1

How Long?

- $1\ 2\ 3\ \cdots\ n$: LIS has length n
- $n\ n-1\ \cdots 2\ 1$: LIS has length 1
- What is typical?

How Long?

- $1\ 2\ 3\ \cdots\ n$: LIS has length n
- $n\ n-1\ \cdots 2\ 1$: LIS has length 1
- What is typical?
- (Ulam, 1960's) Given a (uniformly) random arrangement of $1, 2, \ldots, n$ what is the expected length μ_n of its longest increasing subsequence?

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}}$$

exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n \to \infty} \frac{\mu_n}{\sqrt{n}}$$

exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.

- Erdös and Szekeres (1935): "proved" $K_{LIS} \geq 1/2$

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}}$$

  exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.
- Erdös and Szekeres (1935): "proved" $K_{LIS} \geq 1/2$
- Baer and Brock (1968): guessed $K_{LIS} = 2$

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}}$$

  exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.
- Erdös and Szekeres (1935): "proved" $K_{LIS} \geq 1/2$
- Baer and Brock (1968): guessed $K_{LIS} = 2$
- Hammersley (1972): proved K_{LIS} exists

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}}$$

  exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.
- Erdös and Szekeres (1935): "proved" $K_{LIS} \geq 1/2$
- Baer and Brock (1968): guessed $K_{LIS} = 2$
- Hammersley (1972): proved K_{LIS} exists
- Logan and Shepp (1977): proved $K_{LIS} \geq 2$

The expected length μ_n of the LIS: history

- Ulam conjectured that

$$K_{LIS} = \lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}}$$

  exists; i.e. $\mu_n \sim K_{LIS}\sqrt{n}$.
- Erdös and Szekeres (1935): "proved" $K_{LIS} \geq 1/2$
- Baer and Brock (1968): guessed $K_{LIS} = 2$
- Hammersley (1972): proved K_{LIS} exists
- Logan and Shepp (1977): proved $K_{LIS} \geq 2$
- Vershik and Kerov (1977): proved $K_{LIS} = 2$

The expected length μ_n of the LIS: recent

- Baik, Deift, and Johansson (1999): found the complete limiting distribution in terms of the *Tracey-Widom* distribution. From this can be computed the higher moments including the standard deviation σ_n.

The expected length μ_n of the LIS: recent

- Baik, Deift, and Johansson (1999): found the complete limiting distribution in terms of the *Tracey-Widom* distribution. From this can be computed the higher moments including the standard deviation σ_n.
-

$$\begin{aligned} \mu_n &= 2n^{1/2} - \gamma n^{1/6} + o(n^{1/6}) \\ \sigma_n &= \delta n^{1/6} + o(n^{1/6}) \end{aligned}$$

where $\gamma = 1.711$ and $\delta = 0.902$.

The LIS algorithm: Schensted (1961)

- Scan from left to right

The LIS algorithm: Schensted (1961)

- Scan from left to right
- Maintain "best" IS's of each length

The LIS algorithm: Schensted (1961)

- Scan from left to right
- Maintain "best" IS's of each length
- Update the best IS's after reading each term

The LIS algorithm: Schensted (1961)

- Scan from left to right
- Maintain "best" IS's of each length
- Update the best IS's after reading each term
- Ultimately return the longest of the best IS's

The LIS algorithm: Schensted (1961)

- Scan from left to right
- Maintain "best" IS's of each length
- Update the best IS's after reading each term
- Ultimately return the longest of the best IS's

What does "best" mean?
Most easily extended – i.e. having smallest last term.
Often just need the final term of a best IS.

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25											
2												
3												
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25										
2		30										
3												
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18									
2		30	30									
3												
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18								
2		30	30	20								
3												
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18							
2		30	30	20	20							
3					56							
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18						
2		30	30	20	20	20						
3					56	21						
4												
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18					
2		30	30	20	20	20	20					
3					56	21	21					
4							45					
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17				
2		30	30	20	20	20	20	20				
3					56	21	21	21				
4							45	45				
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17	17			
2		30	30	20	20	20	20	20	20			
3					56	21	21	21	21			
4							45	45	40			
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17	17	15		
2		30	30	20	20	20	20	20	20	20		
3					56	21	21	21	21	21		
4							45	45	40	40		
5												

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17	17	15	15	
2		30	30	20	20	20	20	20	20	20	20	
3					56	21	21	21	21	21	21	
4							45	45	40	40	40	
5											43	

Example: LIS of 25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17	17	15	15	15
2		30	30	20	20	20	20	20	20	20	20	16
3					56	21	21	21	21	21	21	21
4							45	45	40	40	40	40
5											43	43

Example: LIS of $25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16$

	25	30	18	20	56	21	45	17	40	15	43	16
1	25	25	18	18	18	18	18	17	17	15	15	15
2		30	30	20	20	20	20	20	20	20	20	16
3					56	21	21	21	21	21	21	21
4							45	45	40	40	40	40
5											43	43

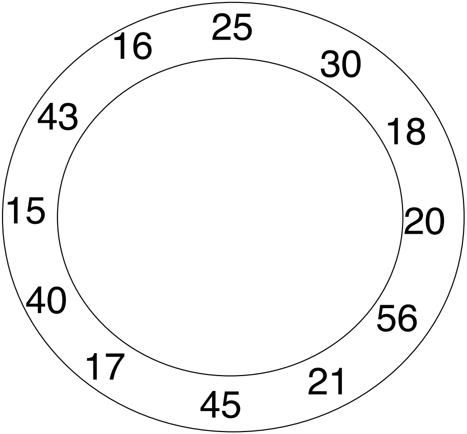
The LIS has length 5 and is $18, 20, 21, 40, 43$

Analysis

- Only store the last column
- New last column obtained from old by binary search and insertion
- Some back pointers to actually construct the result
- Total time $O(n \log n)$

The Longest Increasing Circular Subsequence (LICS)

$25, 30, 18, 20, 56, 21, 45, 17, 40, 15, 43, 16$ in a circle.



The LICS is not $18, 20, 21, 40, 43$ but is $15, 16, 18, 20, 21, 45$

LICS versus LIS

If π is a permutation then

- $|LIS(\pi)| \leq |LICS(\pi)| \leq 2|LIS(\pi)|$
- Upper bound attained for $\pi = k+1, k+2, \ldots, 2k, 1, 2, \ldots, k$
- Lower bound attained for $\pi = 1, 2, \ldots, n$

How do we find the LICS?

- We could run the LIS algorithm on the n different ways of regarding the circular sequence as a linear sequence.

How do we find the LICS?

- We could run the LIS algorithm on the n different ways of regarding the circular sequence as a linear sequence.
- That takes time $O(n^2 \log n)$

How do we find the LICS?

- We could run the LIS algorithm on the n different ways of regarding the circular sequence as a linear sequence.
- That takes time $O(n^2 \log n)$
- Can we do better?

Why would we be interested in the LICS?

Why would we be interested in the LICS?

Recent applications of LIS detection algorithms in matching gene sequences, and some bacteria have apparently circular sequences; special case of general permutation pattern class problems, guff, guff, guff....

Why would we be interested in the LICS?

"Because it's there"

Recent applications of LIS detection algorithms in matching gene sequences, and some bacteria have apparently circular sequences; special case of general permutation pattern class problems, guff, guff, guff....

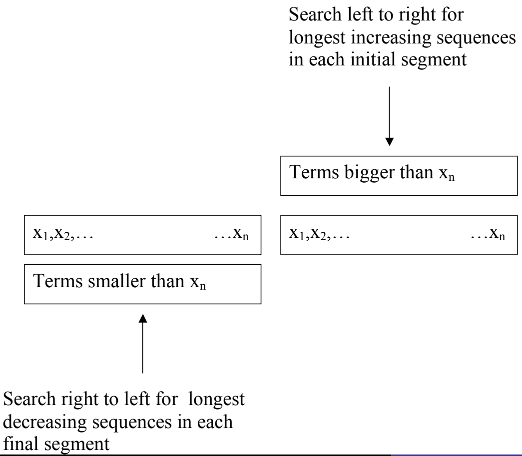
Passing through

Lemma

If $t_1, \ldots, t_m$ is the final column of the LIS algorithm then there is an LICS that contains at least one of these terms.

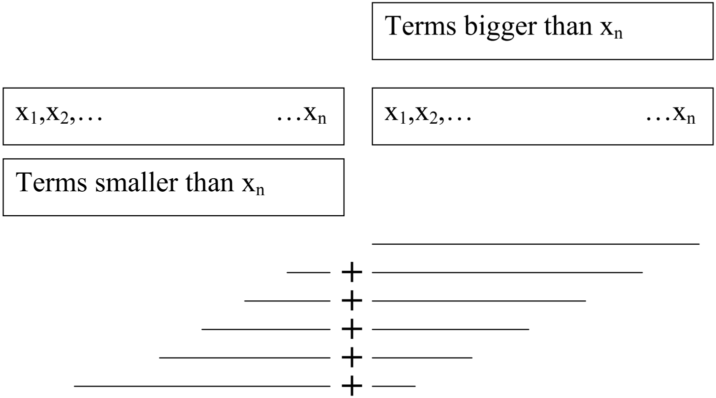
There and back again

We can find the LICS in time $O(n \log n)$ if we know any term that it contains.



There and back again

We can find the LICS in time $O(n \log n)$ if we know any term that it contains.



Algorithm for the LICS

- Run the LIS algorithm and find the last column $t_1, \ldots, t_m$

Algorithm for the LICS

- Run the LIS algorithm and find the last column $t_1, \ldots, t_m$
- Find the longest increasing circular sequence through each t_i ($2m$ applications of LIS algorithm)

Algorithm for the LICS

- Run the LIS algorithm and find the last column $t_1, \ldots, t_m$
- Find the longest increasing circular sequence through each t_i ($2m$ applications of LIS algorithm)
- Return the longest of these

The execution time

Theorem

The expected execution time of the LICS algorithm is $O(n\sqrt{n}\log n)$.

Proof.

The expected value of m is $2\sqrt{n}$. There are $2m+1$ applications of the LIS algorithm each taking time $O(n\log n)$. $\square$

The execution time

Theorem

The expected execution time of the LICS algorithm is $O(n\sqrt{n}\log n)$.

Proof.

The expected value of m is $2\sqrt{n}$. There are $2m+1$ applications of the LIS algorithm each taking time $O(n\log n)$. $\square$

- All the implied O constants are small.

The execution time

Theorem

The expected execution time of the LICS algorithm is $O(n\sqrt{n}\log n)$.

Proof.

The expected value of m is $2\sqrt{n}$. There are $2m+1$ applications of the LIS algorithm each taking time $O(n\log n)$. $\square$

- All the implied O constants are small.
- This is a practical algorithm

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.
  1. Run LIS and get last column $t_1, \ldots, t_m$.

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.
  1. Run LIS and get last column $t_1, \ldots, t_m$.
  2. If $m \leq 3\sqrt{n}$ proceed as above. Otherwise $|LICS(\pi)| \geq |LIS(\pi)| = m > 3\sqrt{n}$

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.
  1. Run LIS and get last column $t_1, \ldots, t_m$.
  2. If $m \leq 3\sqrt{n}$ proceed as above. Otherwise $|LICS(\pi)| \geq |LIS(\pi)| = m > 3\sqrt{n}$
  3. Make random guesses for a term s in $LICS(\pi)$; run "There and back" on such terms.

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.
  1. Run LIS and get last column $t_1, \ldots, t_m$.
  2. If $m \leq 3\sqrt{n}$ proceed as above. Otherwise $|LICS(\pi)| \geq |LIS(\pi)| = m > 3\sqrt{n}$
  3. Make random guesses for a term s in $LICS(\pi)$; run "There and back" on such terms.
  4. Return longest sequence found.

Monte-Carlo variant

- A Monte-Carlo algorithm is one with a small chance of giving the wrong answer; the error probability can be controlled.
- Monte-Carlo algorithm for $LICS(\pi)$ with worst-case time complexity $O(n\sqrt{n}\log n)$.
  1. Run LIS and get last column $t_1, \ldots, t_m$.
  2. If $m \leq 3\sqrt{n}$ proceed as above. Otherwise $|LICS(\pi)| \geq |LIS(\pi)| = m > 3\sqrt{n}$
  3. Make random guesses for a term s in $LICS(\pi)$; run "There and back" on such terms.
  4. Return longest sequence found.
- If we make more than $-\sqrt{n}\ln(\epsilon)/3$ guesses the probability that *none* of them lie in the LICS is less than ϵ.

Lies, damned lies and statistics

10000 random trials gave estimates μ_n-est. for μ_n for various n.

n	10000	12000	14000	16000	18000	20000
μ_n-est.	200.1	219.3	237.0	253.4	268.8	283.4

Lies, damned lies and statistics

10000 random trials gave estimates μ_n-est. for μ_n for various n.

n	10000	12000	14000	16000	18000	20000
μ_n-est.	200.1	219.3	237.0	253.4	268.8	283.4
$2\sqrt{n}$	200.0	219.1	236.7	253.0	268.3	282.8

It appears that μ_n is much closer to $2\sqrt{n}$ than it was in the linear case. But all we can prove is...

The mean of the LICS

Theorem

$$\lim_{n\to\infty} \frac{\mu_n}{\sqrt{n}} = 2.$$