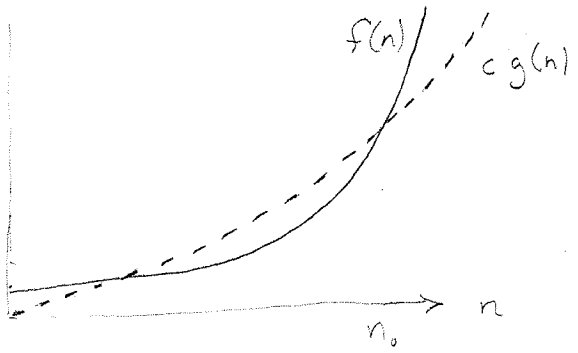
Big Omega notation (refresher)

$$\Omega(g(n)) = \{ f(n) : \exists \text{ positive } c \text{ and } n_0 \text{ such that } f(n) \geq c \cdot g(n) \text{ for all } n > n_0 \}$$



$f(n) \in \Omega(g(n))$

means $g(n)$ is an asymptotic lower bound on $f(n)$.

Is Insertion Sort's running time $\in \Omega(n^2)$?

$$\underbrace{\frac{n^2}{2} + \frac{5n}{2}}_{f(n)} \geq \frac{1}{2} \underbrace{n^2}_{g(n)}$$

$c = 1/2$

$n_0 = 1$

Note:

$\lim_{n \to \infty} f(n) = A$

means

$\forall \varepsilon > 0 \ \exists n_0$ such that

$|f(n) - A| < \varepsilon$

for all $n > n_0$

Big Theta notation

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

Limits

For positive functions $f(n)$ and $g(n)$,

if $A = \lim_{n \to \infty} f(n)/g(n)$ exists then

little-o	$f(n) \in o(g(n))$	if $A = 0$
	$f(n) \in \Theta(g(n))$	if $A \in \mathbb{R}^+$
little-omega	$f(n) \in \omega(g(n))$	if $A = \infty$

o	$<$
O	$\leq$
Θ	$=$
Ω	$\geq$
ω	$>$

Note: $f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$

$f(n) \in \omega(g(n)) \Rightarrow f(n) \in \Omega(g(n))$

Warning: Not every pair of functions are comparable

e.g. n versus $n^{1+\sin n}$

For example: show $\log_2 n \in o(\sqrt{n})$

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \frac{\infty}{\infty} \; ??$$

L'Hopital's Rule

If $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$ and $\lim_{n\to\infty} \frac{f'(n)}{g'(n)}$ exists

then $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$.

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 e)\,1/n}{1/2\, n^{-1/2}} = \lim_{n\to\infty} \frac{2\log_2 e}{\sqrt{n}} = 0$$

Recursion

db skyline
= undominated points

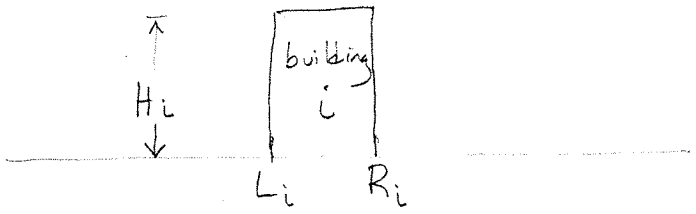
The Skyline Problem

Given a set of n rectangular buildings in a 2 dimensional world

Find the skyline of the buildings eliminating hidden lines.

Input $B[1 \ldots n]$ an array of n buildings where

$B[i] = (L_i \; H_i \; R_i)$ is a triple of real numbers



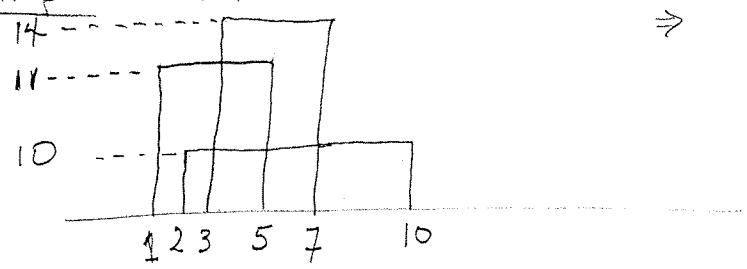
Output $((x_1, y_1)(x_2, y_2) \cdots (x_m, y_m))$ ← skyline

$y_m = 0$

A polygonal line made by connecting

$(x_1, 0)$ to (x_1, y_1) to (x_2, y_1) to (x_2, y_2) to $(x_3, y_2) \cdots$

Example $(1, 11, 5)$ $(3, 14, 7)$ $(2, 5, 10)$



$\Rightarrow$ $((1, 11), (3, 14), (7, 5), (10, 0))$

Imagine lots of buildings.

Idea If I know the skyline for half of the buildings and I know the skyline for the other half as well I can merge the two skylines to create the total skyline

(Other ideas, e.g. input consuming, would work also.)

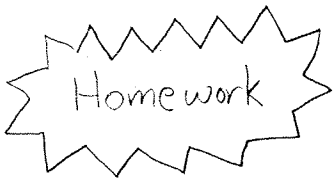
```
Skyline( B[1...n])
    If n=1 then return (L, H, R, 0)
    U = skyline( B[1... ⌈n/2⌉])
    V = skyline (B [⌊n/2⌋+1 ... n])
    return Merge( U, V)
```

Recursion = leap of faith

Resist the temptation to unravel recursion when designing algorithms – it is confusing. Have faith.

Other views
- Code
- Stack of stack frames
- tree of invocations

Merge (U, V) merge two skylines



Idea: produce output from left to right

Skyline can only change at x-coordinates of U or V

Can be done in time

$$O(|U| + |V|)$$

Output skyline has size $\leq |U| + |V|$

Let $T(n)$ be the running time of skyline on inputs of size n.

Recurrence relation:

$$\begin{cases} T(n) = 1 & \text{if } n = 1 \\ T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + c \cdot n & \text{if } n > 1 \end{cases}$$

(c: some positive constant)

Let's assume n is a power of 2 so $\lfloor n/2 \rfloor = \lceil n/2 \rceil = n/2$.
[We'll worry about this later...]

So... $T(1) = 1$
$T(n) = 2T(n/2) + cn$ for n a power of 2.

$T(1) = 1$
$T(2) = 2T(1) + 2c = 2 + 2c = 2(1+c)$
$T(4) = 2T(2) + 4c = 4 + 4c + 4c = 4(1+2c)$
$T(8) = 2T(4) + 8c = 8 + 8c + 8c + 8c = 8(1+3c)$
$T(16) = 2T(8) + 16c = 16 + 16c + 16c + 16c + 16c = 16(1+4c)$

Maybe $T(n) = n(c \log_2 n + 1)$??

Try to prove it... by induction.

Claim $T(n) = n(c \lg n + 1)$ where $T(1) = 1$, $T(n) = 2T(n/2) + cn$
and $n = 2^k$

Proof (by induction on k)

For $k=0$, $n=1$, $T(1) = 1 = 1(c \lg 1 + 1)$ ✓

I.H. Suppose claim is true for $n = 2^k$

I.S. Show it is true for $n = 2^{k+1}$

$$T(2^{k+1}) = 2T(2^k) + c\,2^{k+1}$$
$$= 2\left(2^k (ck+1)\right) + c\,2^{k+1} = 2^{k+1}\left(c(k+1)+1\right)$$ ✓