Useful facts

Order Notation Summary:
$f(n) \in O(g(n))$ if $\exists\ c > 0$ and $n_0 \geq 0$ such that $|f(n)| \leq c\,|g(n)|\ \forall\ n \geq n_0$.
$f(n) \in \Omega(g(n))$ if $\exists\ c > 0$ and $n_0 \geq 0$ such that $|f(n)| \geq c\,|g(n)|\ \forall\ n \geq n_0$.
$f(n) \in \Theta(g(n))$ if $\exists\ c_1, c_2 > 0$ and $n_0 \geq 0$ such that $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|\ \forall\ n \geq n_0$.
$f(n) \in o(g(n))$ if $\forall\ c > 0\ \exists\ n_0 \geq 0$ such that $|f(n)| \leq c\,|g(n)|\ \forall\ n \geq n_0$.
$f(n) \in \omega(g(n))$ if $\forall\ c > 0\ \exists\ n_0 \geq 0$ such that $|f(n)| \geq c\,|g(n)|\ \forall\ n \geq n_0$.

Some useful sums:

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$
- $\sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma + o(1) \in \Theta(\log n)$
- $\sum_{i=0}^{\infty} \frac{1}{2^i} = 2$
- $\sum_{i=0}^{\infty} \frac{i}{2^i} = 2$
- $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$
- $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$

Some well-known sequences:

n	0	1	2	3	4	5	6	7	8	9
Power of 2, 2^n	1	2	4	8	16	32	64	128	256	512
Factorial $n!$	1	1	2	6	24	120	720	5040	40320	362 880
Fibonacci number $F(n)$	0	1	1	2	3	5	8	13	21	34
Catalan-number $C(n)$	1	1	2	5	14	42	132	429	1430	4862

Randomization, probability and moments:

- `random(int` n`)` returns an integer in $\{0, \ldots, n-1\}$, chosen uniformly.
- $E[aX] = aE[X]$, $E[X+Y] = E[X] + E[Y]$ (linearity of expectation)
- $V[X] = V[a+X]$
- Chebyshev's inequality: $P(|X - E[X]| \geq t) \leq \frac{V(X)}{t^2}$

Some recursions that we have seen:

Recursion	resolves to
$T(n) = T(n/2) + \Theta(1)$	$T(n) \in \Theta(\log n)$
$T(n) = 2T(n/2) + \Theta(n)$	$T(n) \in \Theta(n \log n)$
$T(n) = 2T(n/2) + \Theta(\log n)$	$T(n) \in \Theta(n)$
$T(n) = T(cn) + \Theta(n)$ for some $0 < c < 1$	$T(n) \in \Theta(n)$
$T(n) = \frac{1}{2}T(\frac{3}{4}n) + \frac{1}{2}T(n-1) + \Theta(1)$	$T(n) \in \Theta(\log n)$
$T(n) = \frac{1}{n}\sum_{i=0}^{n-1} \max\{T(i), T(n-i-1)\} + \Theta(n)$	$T(n) \in \Theta(n)$
$T(n) = \frac{2}{n}\sum_{i=2}^{n-1} T(i) + \Theta(n)$	$T(n) \in \Theta(n \log n)$
$T(n) = \frac{4}{n}\sum_{i=2}^{n-1} T(i)$	$T(n) \in \Theta(n^3)$
$T(n) = T(\sqrt{n}) + \Theta(\sqrt{n})$	$T(n) \in \Theta(\sqrt{n})$
$T(n) = T(\sqrt{n}) + \Theta(1)$	$T(n) \in \Theta(\log \log n)$

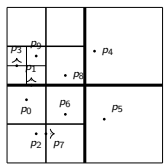
Some definitions that we have seen:

- $F[j]$ = length ℓ of the longest prefix of P that is a suffix of $P[1..j]$
- $F^+[j] = \begin{cases} \text{length } \ell \text{ of the longest prefix of } P \text{ that is a suffix of } P[1..j], \text{ and } P[j{+}1] \neq P[\ell]. \\ 0 \text{ if there is no such } \ell. \end{cases}$
- $L[c] = \begin{cases} \text{maximal index } j \text{ with } P[j] = c \\ -1 \text{ if there is no such } j. \end{cases}$
- $S[j] = \max\ \{\text{index } \ell\text{: } P[j{+}1..m{-}1] \text{ is a prefix of } P^*[\ell{+}1..m{-}2].\}$
- Elias-Gamma code $E(k) = 0^{\lfloor \log k \rfloor} +\!\!+ (k)_2$

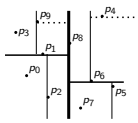
Some slides:

Range search data structures summary

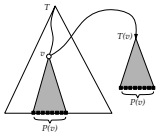
- Quadtrees
  - simple (also for dynamic set of points)
  - work well only if points evenly distributed
  - wastes space for higher dimensions



- kd-trees
  - linear space
  - range search time $O(\sqrt{n} + s)$
  - inserts/deletes destroy balance and range search time (no simple fix)



- range-trees
  - range search time $O(\log^2 n + s)$
  - wastes some space
  - inserts/deletes destroy balance (can fix this with occasional rebuilt)



Convention: Points on split lines belong to right/top side.

String Matching Conclusion

	Brute-Force	Karp-Rabin	DFA	Knuth-Morris-Pratt	Boyer-Moore	Suffix Tree	Suffix Array
Preproc.	—	$O(m)$	$O(m\|\Sigma\|)$	$O(m)$	$O(m{+}\|\Sigma\|)$	$O(n^2\|\Sigma\|)$ [$O(n\|\Sigma\|)$]	$O(n \log n)$ [$O(n)$]
Search time	$O(nm)$	$O(n{+}m)$ expected	$O(n)$	$O(n)$	$O(n)$ or better	$O(m)$	$O(m \log n)$ [$O(m + \log n)$]
Extra space	—	$O(1)$	$O(m\|\Sigma\|)$	$O(m)$	$O(m{+}\|\Sigma\|)$	$O(n)$	$O(n)$

- Our algorithms stopped once they have found one occurrence.
- Most of them can be adapted to find *all* occurrences within the same worst-case run-time.

Compression summary

Huffman	**Run-length encoding**	**Lempel-Ziv-Welch**	**bzip2** (uses Burrows-Wheeler)
variable-length	variable-length	fixed-length	multi-step
single-character	multi-character	multi-character	multi-step
2-pass, must send dictionary	1-pass	1-pass	not streamable
60% compression on English text	bad on text	45% compression on English text	70% compression on English text
optimal 01-prefix-code	good on long runs (e.g., pictures)	good on English text	better on English text
requires uneven frequencies	requires runs	requires repeated substrings	requires repeated substrings
rarely used directly	rarely used directly	frequently used	used but slow
part of pkzip, JPEG, MP3	fax machines, old picture-formats	GIF, some variants of PDF, compress	bzip2 and variants

Extendible hashing

We can save links (hence space in internal memory) with two tricks:

- Expand the trie so that all leaves have the same **global depth** d_D.
- Store *only* the leaves, and in an array D of size 2^{d_D}.