15-210
Parallel and Sequential Algorithms and Data Structures

Lecture 1

Overview – The Genome Sequencing Problem

Major Themes

- Defining precise problem and data abstractions,

- Designing and programming
  - correct and efficient algorithms and data structures
  - for given problems and data abstractions

	Abstraction	**Implementation**
Functions	Problem	Algorithm
Data	Abstract Data Type	Data Structure

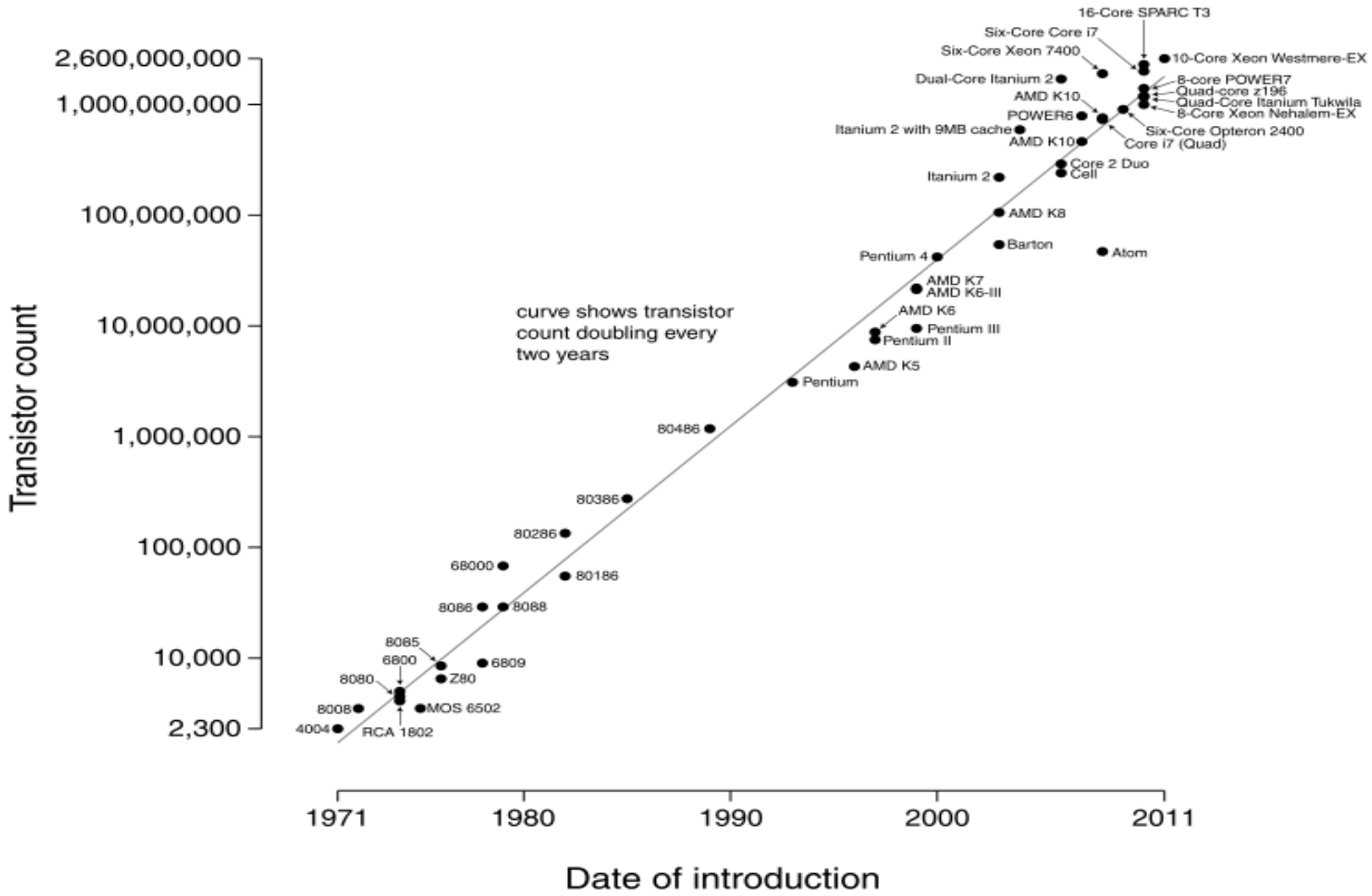
Problem vs. Algorithm

- Sorting, string matching, finding shortest paths in graphs,. . . , are problems
  - **Input**: A sequence $[a_1, a_2, \cdots, a_n]$
  - **Output**: A permutation of the sequence $[a_{i_1}, a_{i_2}, \cdots, a_{i_n}]$ such that $\forall j, 1 \leq j < n, a_{i_j} \leq a_{i_{j+1}}$
- Quicksort, Mergesort, Insertion Sort , . . . , are algorithms for sorting.

Abstract Data Types vs. Data Structures

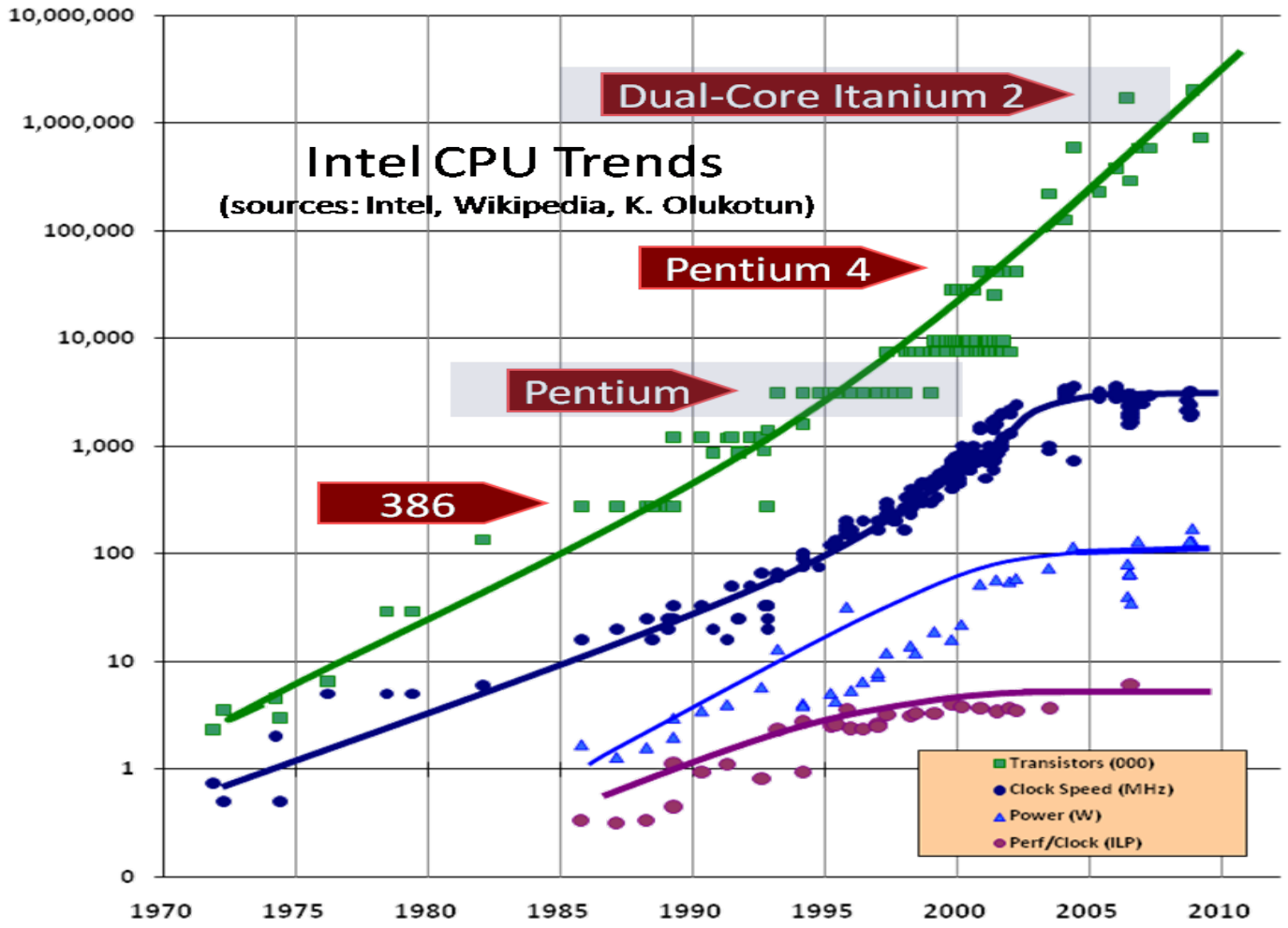
- A set is an abstract data type (ADT)
  - Test membership, intersect, union, difference, . . .
- Sequences, trees, hash-tables are examples of data structures.
- ADT's determine functionality, data structures determine costs.

Technology – Moore's Law

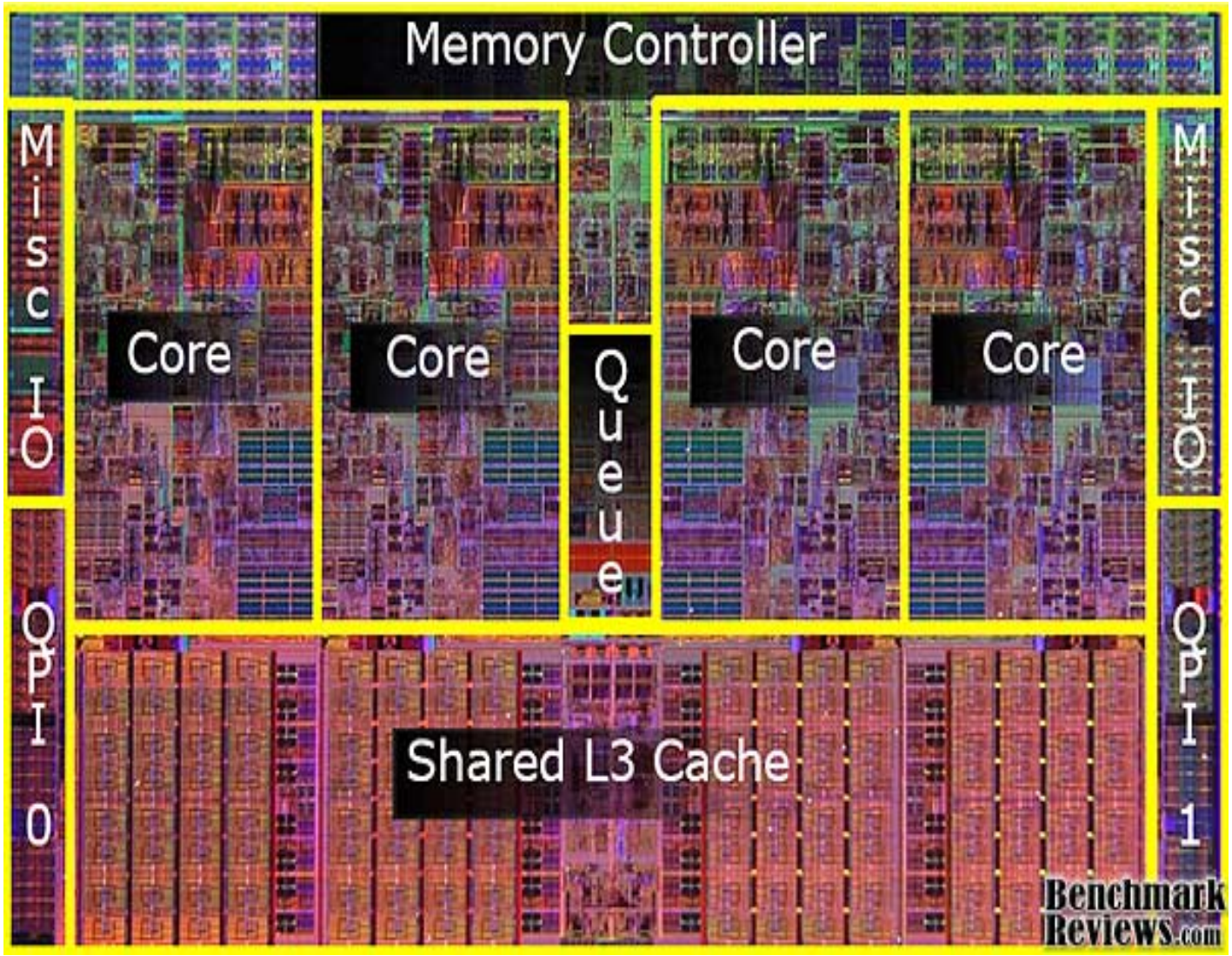


Source: Wikipedia

Processor Technology



Multi-core Chips



Multi-core Chips

Intel Core i7 Processor Series Features & Specifications			
	Intel Core i7-965 Extreme Edition	**Intel Core i7-940**	**Intel Core i7-920**
Clock Speed (GHz)	3.20	2.93	2.66
QPI Speed (GT/sec)	6.4	4.8	4.8
Socket	1366-pin LGA		
Cache	8 Megabytes		
Memory Speed Support	DDR3-1066		
TDP	130 Watts		
Overspeed Protection Removed	Yes	No	No
Processor Architecture	New Intel Core micro architecture (Nehalem) 45nm		
Key Platform Features	• Intel Hyper-Threading Technology delivers 8-threaded performance on 4 cores • Intel Turbo Boost Technology • 8M Intel Smart Cache • Integrated Memory Controller with support for 3 channels of DDR3 1066 memory • Intel QuickPath interconnect to Intel X58 Express Chipset		

Parallel Algorithms

	Serial	Parallel		
		1-core	8-core	32h-core
Sorting 10M strings	2.90	2.90	0.40	.095 (30.5)
Remove dupl. 10M strings	0.66	1.00	0.14	.038 (17.4)
Min. span. tree 10M edges	1.60	2.50	0.42	.140 (11.4)
BFS 10M edges	0.82	1.20	0.20	.046 (17.8)

Running times in seconds

15-210 vs. a Traditional Course

- Emphasis on parallel thinking at a high level
  - Parallel algorithms and parallel data structures
- Purely functional model of computation
  - Safe for parallelism
  - Higher level of abstraction
- Ideas still relevant for imperative computation
  - Lot of overlap, but covered differently!

Synopsis

- A real world problem: Gene sequencing.
- The computational problem.
- Algorithms

Sequencing the Genome

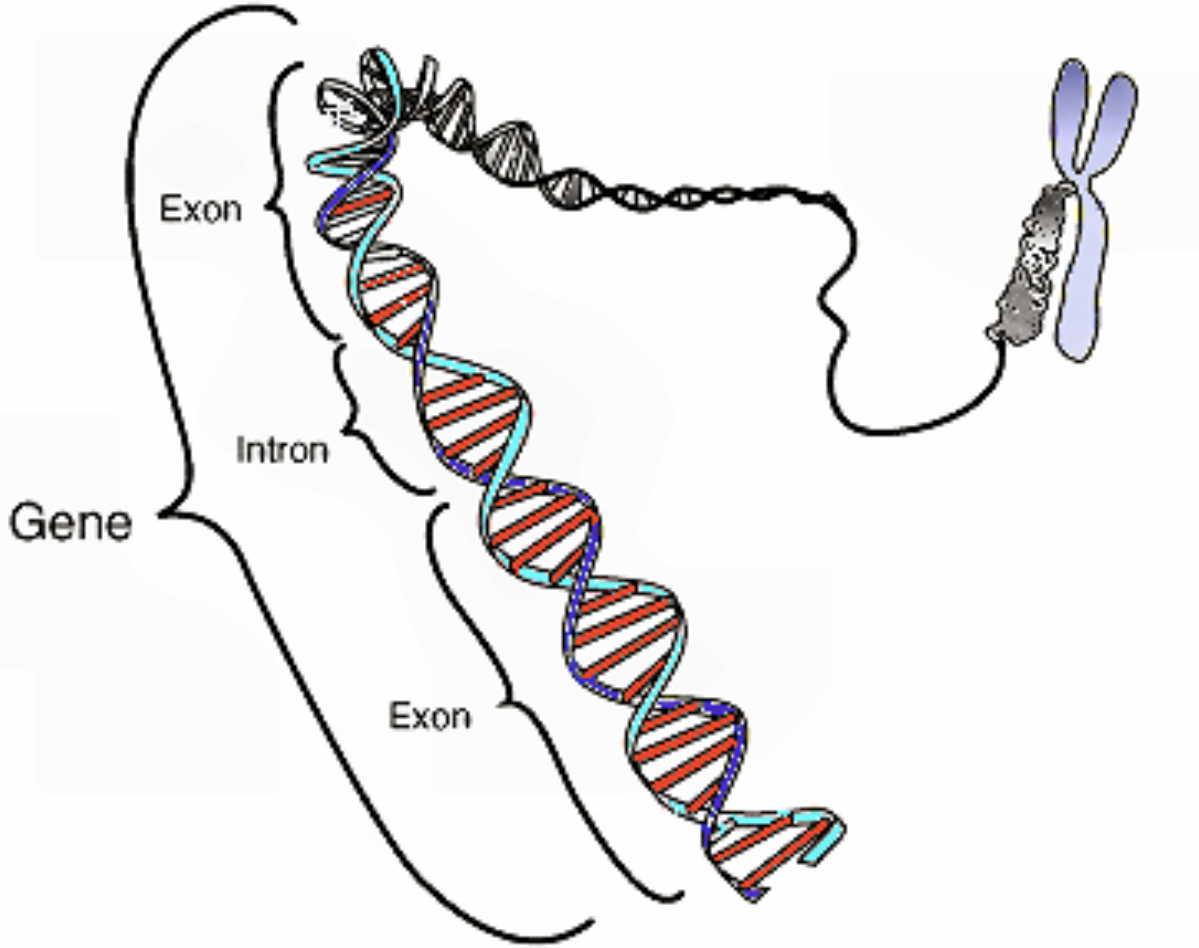
- The human DNA molecule encodes the complete set of genetic information using 4 bases
  - Adenine (A), Cytosine (C), Guanine (G) and Thymine (T)
- A sequence of about
  - 3 billion base pairs
  - arranged into 46 chromosomes

  makes up the human genome.

Sequencing the Genome

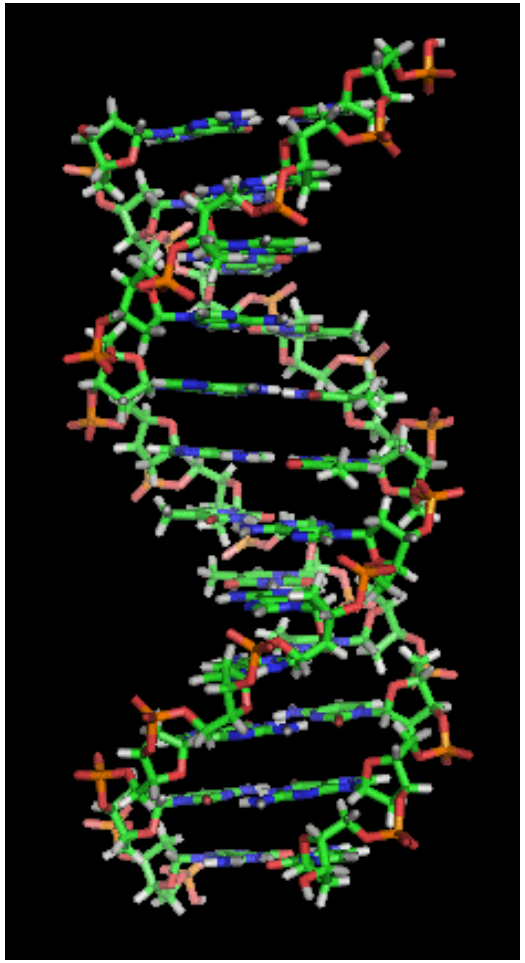
- A chromosome is a sequence of genes
- A gene is a sequence of the base pairs
  - But there seem to be a lot of base-pairs with no apparent functions.

Sequencing the Genome



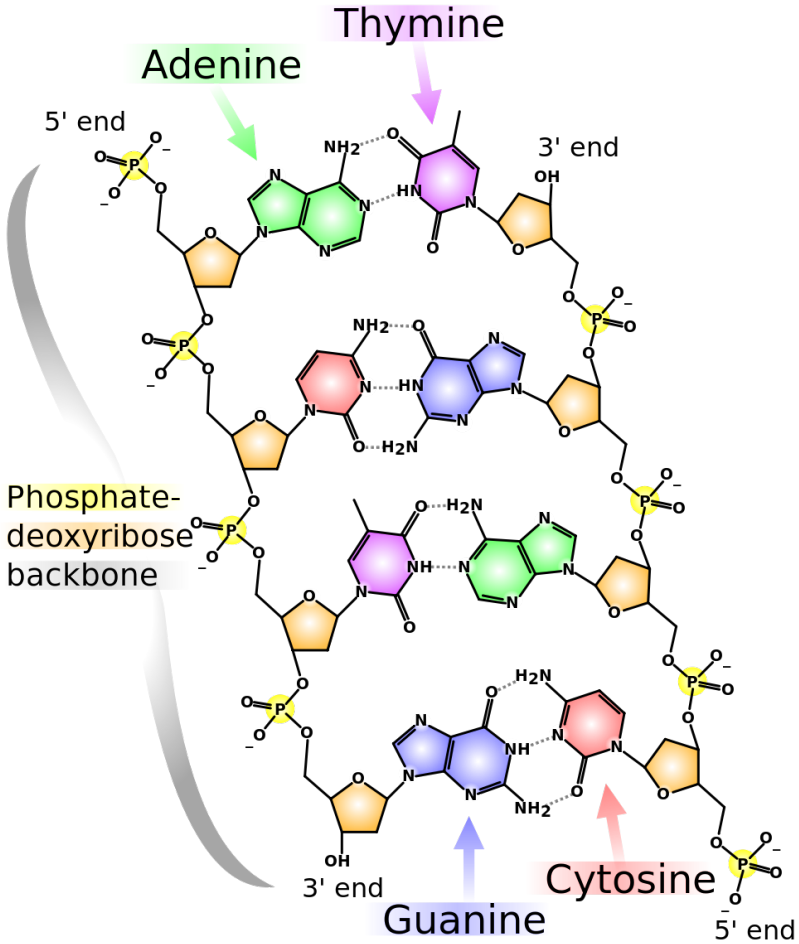
Source: Wikipedia

Sequencing the Genome



Source: Wikipedia

Sequencing the Genome



Source: Wikipedia

Sequencing the Genome

- Determining the complete DNA sequence is a grand challenge.
- Very hard to do in one go with wet lab techniques.
- The Shotgun Technique has been found work quite well.

Shotgun Sequencing

- Break up multiple DNA strands into short segments
  - Chemistry!
- Short segments are sequenced.
  - Chemistry!
- Stitch short sequences computationally.
  - This is where CS comes in.

Shotgun Sequencing

Strand	Sequence
Original	AGCATGCTGCAGTCATGCTTAGGCTA
First shotgun sequence	AGCATGCTGCAGTCATGCT------- --------------------TAGGCTA
Second shotgun sequence	AGCATG-------------------- ------CTGCAGTCATGCTTAGGCTA
Reconstruction	AGCATGCTGCAGTCATGCTTAGGCTA

Source: Wikipedia

Shotgun Sequencing

- Suppose you have three strands sequenced

```
catt ag gagtat
cat tagg ag tat
ca tta gga gtat
```

- But they really come in a messy way, e.g.,

```
catt ag tta cat tagg ag gagtat
tat ca gga gtat
```

- So how do we stitch them?
  - Given a set of overlapping genome subsequences, construct the "best" sequence that includes them all.

Synopsis

- A real world problem: Gene sequencing.
- The computational problem.
- Algorithms

The Abstract Problem

The Shortest Superstring Problem

Given

- an alphabet of symbols Σ, and
- a set of finite strings $S \subseteq \Sigma^+$,

return

- a shortest string r that contains every $s \in S$ as a substring of r.

- Σ, Σ^+
- $\Sigma = \{A, C, G, T\}$

Some Observations

- Ignore strings that are already in other strings. Why?

$$\{\text{catt}, \text{ag}, \text{gagtat}, \text{cat}, \text{tagg}, \text{ag}, \text{tat}, \text{ca}, \text{tta}, \text{gga}, \text{gtat}\}$$

$$\Downarrow$$

$$\{\text{catt}, \text{gagtat}, \text{tagg}, \text{tta}, \text{gga},\}$$

- Each string must start at a distinct position in the result. Why?

Synopsis

- A real world problem: Gene sequencing.
- The computational problem.
- Algorithms:
  - The Brute Force Algorithm

The Brute Force Algorithm

The Brute Force Technique

Enumerate all possible candidate solutions for a problem

- score each solution, and/or
- check each satisfies the problem constraints

Return the best solution.

- How does this apply to the SS Problem?
  - Generate permutations
  - Remove overlaps
  - Stitch strings
  - Select the shortest resulting string

The Brute Force Algorithm

- `catt tta tagg gga gagtat`
- `catt tta tagg gga gagtat`
- `cattaggagtat`

LEMMA

Given a finite set of strings $S \subseteq \Sigma^+$, the brute force technique finds the shortest superstring.

- See handout.

- So what is the problem with this technique?

The Brute Force Algorithm

- There are just too many permutations!
- So, $n = 100 \rightarrow 100! \approx 10^{158}$ permutations.
- Testing at 10^{10} permutations/sec, you need
  - $\approx 10^{148}$ seconds
  - $\approx 10^{143}$ days ($\approx 10^5$ seconds/day)
  - $\approx 2.7 \times 10^{140}$ years
  - $\approx 2.7 \times 10^{138}$ centuries
- Not bloody likely you will test each permutation before hell freezes over!
  - Even if every subatomic particle in the universe was a processor

Prospects for a Faster Algorithm?

- SS belongs to very important class of problems called NP (for Nondeterministic Polynomial).
- For such problems, no algorithm with polynomial work is known.
- But solutions can be verified in polynomial work!
- Wait for 15-451 and 15-453 for the gory details!
- But usually there are approximation algorithms
  - with bounds on the quality of results, and
  - perform better in practice.

Synopsis

- A real world problem: Gene sequencing.
- The computational problem.
- Algorithms:
  - The Brute Force Algorithm
  - Reducing SS to TSP

Problem Reduction

- A reduction is a mapping from one problem (A) to another problem (B), so that the solution B problem can be used to solve A.
  - Solving a set of linear equations, reduces to inverting a matrix.
- Map the instance of problem A to an instance of B,
- Solve using algorithms for B
- Map the resulting solution back.

Reducing SS to TSP

The (Asymmetric) Traveling Salesperson Problem (TSP)

Given a weighted directed graph

- find the shortest path that starts at vertex s, and
- visits each vertex once, and
- returns to s.

- $\equiv$ Hamiltonian path with the lowest total sum of weights
- So, how is this related to SS?

Reducing SS to TSP

- If s_i is followed by s_j in how much will the SS length increase?
  - $s_i = tagg$ followed by $s_j = gga \quad \rightarrow tagg\underline{a}$

- General case?
  - $w_{i,j} = |s_j| - overlap(s_i, s_j)$
  - $overlap(\text{"tagg"}, \text{"gga"}) = 2$
  - $|\text{"gga"}| - 2 = 1$

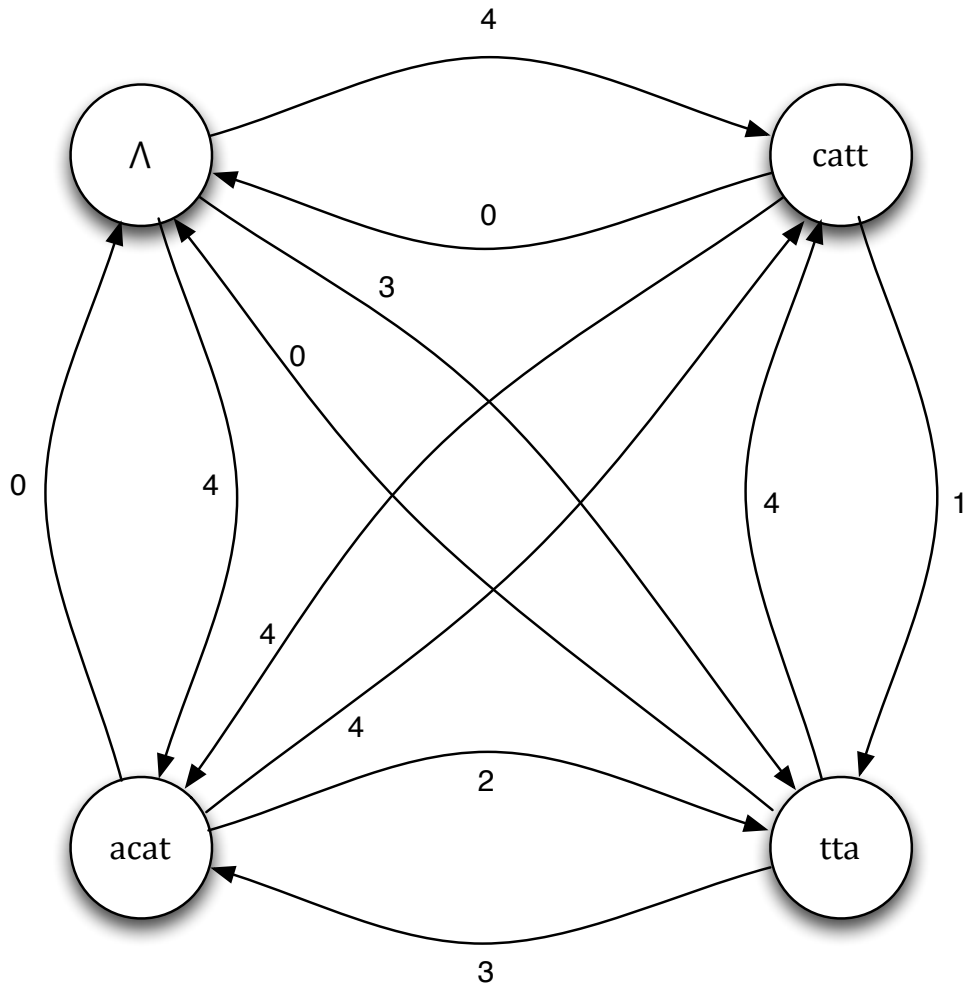
Reducing SS to TSP

Build a graph $D = (V, A)$

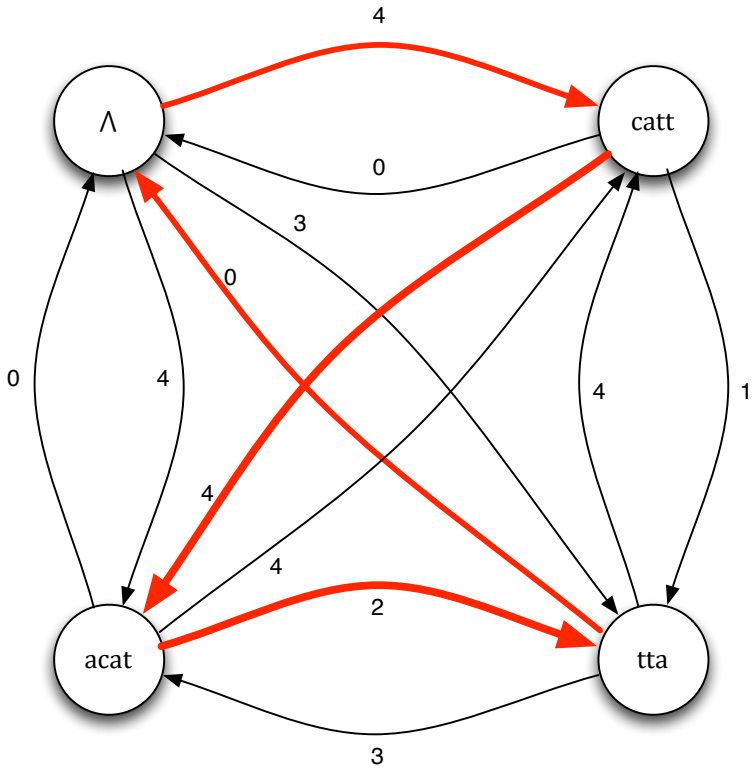
- One vertex for each s_i and one for special "null" node, Λ
- A directed edge from s_i to s_j has weight $w_{i,j} = |s_j| - \mathit{overlap}(s_i, s_j)$
- $w_{\Lambda,i} = |s_i| \rightarrow$ no overlap, maximal increase
- $w_{i,\Lambda} = 0 \rightarrow$, no overlap, no increase

Reducing SS to TSP

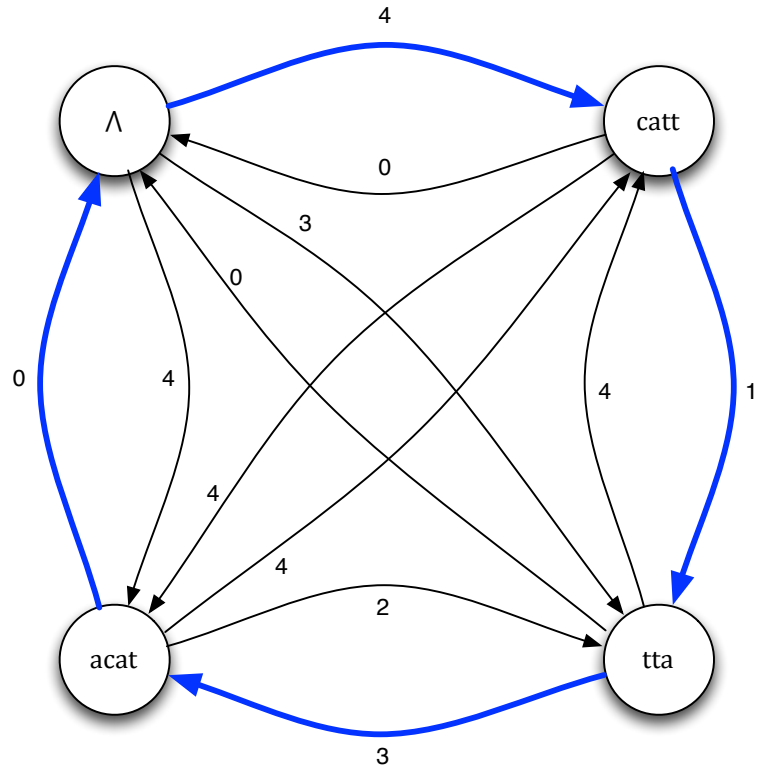
- $S = \{\texttt{catt}, \texttt{tta}, \texttt{acat}\}$



Reducing SS to TSP



- This tour ≡ cattacatta
- Length 10



- This tour ≡ cattacat
- Length 8

Reducing SS to TSP

- TSP considers all Hamiltonian paths (hence is brute force)
- TSP finds the minimum cost Hamiltonian path.
  - Total cost is the length of the SS
- TSP is also NP-hard.

Synopsis

- A real world problem: Gene sequencing.
- The computational problem.
- Algorithms:
  - The Brute Force Algorithm
  - Reducing SS to TSP
  - The Greedy Algorithm

The Greedy Technique

The Greedy Technique

Given a sequence of steps to be made, at each decision point

- make a locally optimal decision
- without ever backtracking on previous decisions.

- Greedy is a quite general algorithmic paradigm.
- In general, it does not get the best solution.
  - But it does work for some other problems (e.g., Huffman Encoding, MST)

The Greedy Approximation to SS

- Start with a pair of strings with maximal overlap (Why?)
- Continue with strings that adds the least extension every time.
  - This is the locally optimal decision!
  - We already defined $overlap(s_i, s_j)$
  - $join(s_i, s_j) \equiv$ concatenate s_j to s_i and remove overlap.
    - $join(\text{"tagg"}, \text{"gga"}) = \text{"tagga"}$

The Greedy Approximation to SS

greedyApproxSS

```
1   fun greedyApproxSS(S) =
2     if |S| = 1 then s_0
3     else let
4         O = {(overlap(s_i, s_j), s_i, s_j) : s_i ∈ S, s_j ∈ S, s_i ≠ s_j}
5         (o, s_i, s_j) = maxval <_#1 O
6         s_k = join(s_i, s_j)
7         S' = ({s_k} ∪ S)\{s_i, s_j}
8     in
9       greedyApproxSS(S')
10    end
```

- `S'` gets smaller by one string after each recursion.

The Greedy Approximation to SS

- `GreedyApproxSS` returns a string with length within 3.5 times the shortest string.
- Conjectured to return within a factor of 2.
- Does much better in practice.

The Greedy Approximation to SS

- Let's do an example.
- $S = \{\text{catt}, \text{gagtat}, \text{tagg}, \text{tta}, \text{gga},\}$

Summary

- Interfaces vs Implementations
  - Precise interfaces are key.
- The Shortest Superstring Problem
  - The brute-force approach
  - Reduction to TSP
  - Approximate solution using greedy paradigm