# Lec 1

Instructors: Guy Blelloch + Charlie Garrod
Today: Motivation for course content
Platform: Diderot
Lab credit caps at 80%
3 exams

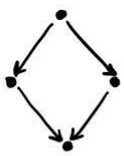
## # Deconstructing course title

### Parallel

- Parallel
- Sequential – special case of parallel by having $n = 1$

Not using multiple cores → wasting your time
Many algorithms inherently parallel → use more cores

### Dependency graph



← Recall work and span
total computation longest path

### Data structure & Algorithm

- Math — Calculus, series, probability, linear algebra, proofs
- Abstraction — Algorithm, interfaces, graphs, asymptotic analysis
- ~~Python~~ — Toolbox ↔ connections → problem, search for solution

Problem solving

connections: recognise similarity btwn problems
search for solution: trial and error with intuition

## # Example problem solving

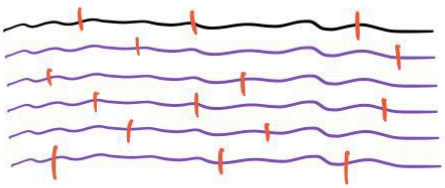
**Problem**: human genome sequencing (2001, 3.1 billions of nucleotide)

String of {A, C, G, T}, 3.1 billion in length

Constraints
- Can't read more than 2000 base pairs
- Sequential read takes 100s of years

→ Technique – Shotgun Method

make multiple copies
shatter into fragments } Done in lab
read each fragment — ~1000 long
reconstruct whole sequence } done in computer
↑ try find overlaps and combine



The algorithm

Get set of all sequences read
Get rid of sequences that are subset of another
Find best reconstruction
↑ Heuristic : find shortest superstring

Reduced problem : Shortest Substring (SS) Problem
└ Also good to check if sth is NP hard └ NP hard!
Informally : given set of strings, find shortest superstring that includes all

Problem solving

→ First try brute force solution, as long as correct
try all permutations, merge overlaps, pick shortest
Correct, but $O(n!)$

→ SS NP hard but has polynomial time approximation and not all possible input instances are hard
Connection : Travelling Salesman Problem ( given graph and distances in edges, visit all nodes with lowest distance )
Reduction : String → vertex
$w(S_1, S_2)$ → - overlap $(S_1, S_2)$
add special vertex $\Lambda$, make $w(S_1, \Lambda) = w(\Lambda, S_1) = 0$
for all $S_1$, to fix cycles