CS481: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/
CS481

- **Class hours:**
  - Online-only weeks:
    - Tue 9:00-10:00 - Thu 15:30-17:20
  - Hybrid weeks:
    - Tue 9:30-10:20 (in class) - Thu 17:30-19:20 (online)

- **Class room:** EB104 / Zoom

- **Office hour:** Wed 14:00-15:00
  - meet.google.com/nhm-ieor-qke

- **TA:** Ricardo Román Brenes: ricardo@bilkent.edu.tr

- **Grading:**
  - 1 midterm: 25%
  - 1 final: 35%
  - Homeworks (programming): 40% (n=7-8)
Recommended Textbooks

- Genome Scale Algorithm Design, Veli Makinen, et al., Cambridge University Press, 2015
- An Introduction to Bioinformatics Algorithms (Computational Molecular Biology), Neil Jones and Pavel Pevzner, MIT Press, 2004
- https://www.bioinformaticsalgorithms.org/

Additional:

- Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Dan Gusfield, Cambridge University Press
- ROSALIND problem sets: http://rosalind.info/problems/locations/
This course is about **algorithms** in the field of bioinformatics:

- What are the problems?
- What algorithms are developed for what problem?
- Algorithm design techniques

This course is **not** about how to analyze biological data using available tools:

- **Recommended course:** MBG 326: Introduction to Bioinformatics
CS481 and other courses

- Includes elements from:
  - CS201/202: data structures -- implicit prerequisite
  - CS473: algorithms, dynamic programming, greedy algorithms, branch-and-bound, etc.
  - CS476: complexity, context-free grammars, DFA/NFA
  - CS464: hidden Markov models (not covered in CS481, but related topic)
CS481: Assumptions

- You are assumed to know/understand
  - Computer science basics (CS101/102 or CS111/112)
    - CS201/202 would be better
    - CS473 would be even better
  - Data structures (trees, linked lists, queues, etc.)
  - Elementary algorithms (sorting, hashing, etc.)
  - Programming: C, C++ (preferred); Python, Java
    - Note: we will give bonus points for the “fastest” code in some homeworks
- You don’t have to be a “biology expert” and we will not teach any biology in this course: MBG 110 would be sufficient
Bioinformatics Algorithms

- Development of methods based on computer science for problems in biology and medicine
  - Sequence analysis (combinatorial and statistical/probabilistic methods)
  - Graph theory
  - Data mining
  - Database
  - Statistics
  - Image processing
  - Visualization
  - .....

CS 481
Bioinformatics: Applications

- Human disease
  - Personalized Medicine
- Genomics: Genome analysis, gene discovery, regulatory elements, etc.
- Population genomics
- Evolutionary biology
- Proteomics: analysis of proteins, protein pathways, interactions
- Transcriptomics: analysis of the transcriptome (RNA sequences)
- ...

[43x475]Bioinformatics: Applications
Why would you learn these algorithms?

- Most developed for research within other fields that include string processing, clustering, text-pattern search, etc.
- Bioinformatics (non-academic) jobs on the rise:
  - Genomics England, Genome Asia, etc.: 100,000 genome projects
  - DNAnexus, SevenBridges, LifeBit: genome analysis on the cloud.
Genomics and healthcare

Stark et al., AJHG 2019
(VERY) BRIEF INTRODUCTION TO COMPLEXITY
Tractable vs intractable

- Tractable problems: there exists a solution with \( O(f(n)) \) run time, where \( f(n) \) is polynomial
- \( P \) is the set of problems that are known to be solvable in polynomial time
- \( NP \) is the set of problems that are verifiable in polynomial time (or, solvable by a non-deterministic Turing Machine in polynomial time)
  - \( NP \): “non-deterministically polynomial” \( P \subseteq NP \)
NP-hard

- **NP-hard**: non-deterministically polynomial - hard
  - Set of problems that are “at least as hard as the hardest problems in NP”
  - There are no known polynomial time optimal solutions
  - There *may* be polynomial-time *approximate* solutions
NP-Complete

A decision problem $C$ is in NPC if:

- $C$ is in NP
- Every problem in NP is reducible to $C$ in polynomial time

That means: if you could solve any NPC problem in polynomial time, then you can solve all of them in polynomial time.

*Decision problems*: outputs “yes” or “no”
NP-intermediate

- Problems that are in NP; but not in either NPC or NP-hard (as far as we know)
P vs. NP

- We do not know whether $P=NP$ or $P\neq NP$
  - Principal unsolved problem in computer science
  - Most likely $P\neq NP$
P vs. NP vs. NPC vs. NP-hard

- P ≠ NP
- P = NP
- P = NP = NP-Complete
- NP-Hard
- NP-Complete
- NP
- P
Examples

- **P:**
  - Sorting numbers, searching numbers, pairwise sequence alignment, etc.

- **NP-complete:**
  - Subset-sum, traveling salesman, etc.

- **NP-intermediate:**
  - Factorization, graph isomorphism, etc.
Historical reference

- The notion of NP-Completeness: Stephen Cook and Leonid Levin independently in 1971
  - First NP-Complete problem to be identified: Boolean satisfiability problem (SAT)
    - Cook-Levin theorem
- More NPC problems: Richard Karp, 1972
  - “21 NPC Problems”
- Now there are thousands….
Sample problem: Change

- **Input:** An amount of money $M$, in cents
- **Output:** Smallest number of coins that adds up to $M$
  - Quarters (25c): $q$
  - Dimes (10c): $d$
  - Nickels (5c): $n$
  - Pennies (1c): $p$
  - Or, in general, $c_1$, $c_2$, ..., $c_d$ ($d$ possible denominations)
Algorithm design techniques

- Exhaustive search / brute force
  - Examine every possible alternative to find a solution

```plaintext
BRUTEFORCECHANGE(M, c, d)
1  smallestNumberOfCoins ← ∞
2  for each (i₁, . . . , iₙ) from (0, . . . , 0) to (M/c₁, . . . , M/cₙ)
3      valueOfCoins ← \sum_{k=1}^{d} i_k c_k
4      if valueOfCoins = M
5          numberOfCoins ← \sum_{k=1}^{d} i_k
6          if numberOfCoins < smallestNumberOfCoins
7              smallestNumberOfCoins ← numberOfCoins
8          bestChange ← (i₁, i₂, . . . , iₙ)
9  return (bestChange)
```
Algorithm design techniques

- **Greedy algorithms:**
  - Choose the “most attractive” alternative at each iteration

```c
 BETTERCHANGE(M, c, d)
 1  r ← M
 2  for k ← 1 to d
 3     i_k ← r/c_k
 4     r ← r - c_k · i_k
 5  return (i_1, i_2, ..., i_d)

 USCHANGE(M)
 1  r ← M
 2  q ← r/25
 3  r ← r - 25 · q
 4  d ← r/10
 5  r ← r - 10 · d
 6  n ← r/5
 7  r ← r - 5 · n
 8  p ← r
 9  return (q, d, n, p)
```
Algorithm design techniques

- **Dynamic Programming:**
  - Break problems into subproblems; solve subproblems; merge solutions of subproblems to solve the real problem
  - Keep track of computations to avoid recomputing values that you already solved
    - *Dynamic programming table*
DP example: Rocks game

- Two players
- Two piles of rocks with $p_1$ rocks in pile 1, and $p_2$ rocks in pile 2
- In turn, each player picks:
  - One rock from either pile 1 or pile 2; OR
  - One rock from pile 1 and one rock from pile 2
- The player that picks the last rock wins
DP algorithm for Player 1

- Problem: \( p_1 = p_2 = 10 \)
- Solve more general problem of \( p_1 = n \) and \( p_2 = m \)
- It’s hard to directly calculate for \( n=5 \) and \( m=6 \); we need to solve smaller problems
### DP algorithm for Player 1

<table>
<thead>
<tr>
<th>pile2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initialize; obvious win for Player 1 for 1,0; 0,1 and 1,1
DP algorithm for Player 1

<table>
<thead>
<tr>
<th>pile2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Player 1 cannot win for 2,0 and 0,2
DP algorithm for Player 1

<table>
<thead>
<tr>
<th>pile2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Player 1 can win for 2,1 if he picks one from pile2

Player 1 can win for 1,2 if he picks one from pile1
DP algorithm for Player 1

Player 1 can win for 2,1 if he picks one from pile2

Player 1 can win for 1,2 if he picks one from pile1
DP algorithm for Player 1

<table>
<thead>
<tr>
<th>pile2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>pile1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Player 1 cannot win for 2, 2

Any move causes his opponent to go to W state
DP “moves”

When you are at position \((i,j)\)

Go to:

Pick from pile 1: \((i-1, j)\)

Pick from pile 2: \((i, j-1)\)

Pick from both piles 1 and 2: \((i-1, j-1)\)
## DP final table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>3</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>7</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>9</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
</tr>
</tbody>
</table>

Also keep track of the choices you need to make to achieve W and L states:  *traceback table*
Algorithm design techniques: CS473

- **Branch and bound:**
  - Omit a large number of alternatives when performing brute force

- **Divide and conquer:**
  - Split, solve, merge
    - Mergesort

- **Machine learning (CS 464):**
  - Analyze previously available solutions, calculate statistics, apply most likely solution

- **Randomized algorithms:**
  - Pick a solution randomly, test if it works. If not, pick another random solution