1MB304: Discrete structures for bioinformatics II

(or Algorithms for bioinformatics)

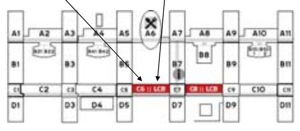
Torgeir R. Hvidsten

Research interests

- ➢ Machine learning in bioinformatics
- ➢ Protein structure prediction
- ➢ Protein-drug interactions
- ➢ Gene regulation

Lecturer:
Torgeir R. Hvidsten
Assistant professor in Bioinformatics
Umeå plant science center
torgeir.hvidsten@plantphys.umu.se

Assistant (responsible for exercises) :
Feifei Xu
The Linnaeus centre for bioinformatics
xffhello@gmail.com



Course goals

After this course you should be able to:

1. describe the different algorithm design techniques and discuss their pros and cons.
2. sketch a solution to a bioinformatics problem using pseudo-code and analyze its time/space complexity
3. recognize the algorithm design technique used in an existing bioinformatics solution, analyze its time/space complexity and the plausibility of using other techniques.
4. translate a given a biological problem into a representation that lends itself to be solved by one of the techniques, and discuss/argue for your solution.

Course information (I)

- ➢ Book: Jones and Pevzner. *An introduction to bioinformatics algorithms*, ISBN 0-262-10106-8 (available at *Akademibokhandeln*).
- ➢ Credit points: 5 (4 points: Exam, 1 point: hand-ins/project)
- ➢ Obligatory hand-in exercises (4 of 6 exercises must be returned and approved)
- ➢ One obligatory computer project including a written report, a literature study, an oral presentation and student review (students may work in pairs)
- ➢ Bonus points on the exam:
  - Up to one point for each approved exercise handed in within the deadlines
  - Up to four bonus points if the project is approved and handed in within the deadline
  - A maximum of 10 bonus points amounting to 10% of the exam

Student correction of exercises

- ➢ For each exercise:
  - One week to hand in a copy of your answers
  - Another week to correct your answers using my suggestions to solutions
- ➢ To get bonus points:
  - Meet both deadlines
  - At least 50% correct
  - All tasks answered *and* corrected
- ➢ Why? It gives you:
  - A second chance to learn the material
  - The opportunity to understand someone else's answers
  - A chance to view your answers in the light of someone else's answers
- ➢ No, it's not less work for the teacher
- ➢ No, it's not more work for you (one topic/lecture less than last year)

Course information (II)

- ➢ Course webpage:
  - http://www.trhvidsten.com/DSB/
- ➢ Here you can find the
  - course program
  - deadlines
- ➢ and download
  - lecture slides
  - exercises/project descriptions
  - additional material not in the book

Content (I)

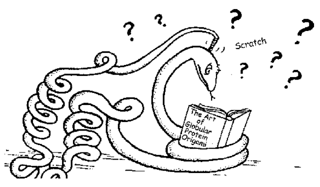
- Exhaustive search (Chapter 4)
  - Application: restriction mapping, finding regulatory motifs in DNA sequences
- Greedy algorithms (Chapter 5)
  - Application: genome rearrangements, finding regulatory motifs in DNA sequences
- Dynamic programming and divide-and-conquer algorithms (Chapters 6, 7.3, 7.4 and 9.8)
  - Sequence alignments (global, local, gaps, multiple alignments)
  - Application: gene prediction, BLAST

Content (II)

- Hidden Markov models (Chapter 11 + research article)
  - Application: Modeling multiple alignments, Pfam
- Randomized algorithms (Chapter 12)
  - Application: Motif finding

Content (III)

- Protein structure prediction from sequence (Project description)
  - Approaches based on fragment libraries
  - The computer project: predicting local structure from sequence



This lecture

- Discrete structures
- Algorithms and pseudo-code
- Algorithm complexity
- Bioinformatics and computational problems in molecular biology

Discrete structures …

- Discrete comes from the Latin word *discretus* which means separate
- Discrete mathematics: branch of mathematics dealing with questions involving finite or countably infinite sets
- In computer science a computation is the progression of a digital computer in a state space as dictated by an algorithm
- Molecular biology: DNA, RNA, proteins, interaction networks, regulatory networks, etc.

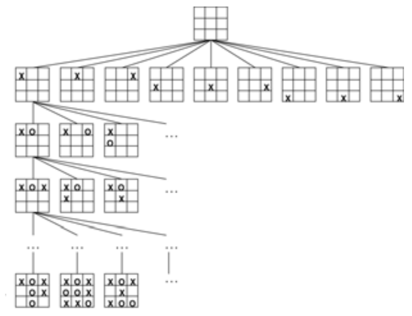
Algorithm

- Algorithm: a sequence of instructions that one must perform in order to solve a well-formulated problem
- Correct algorithm: translate every input instance into the correct output
- Incorrect algorithm: there is at least one input instance for which the algorithm does not produce the correct output
- Many successful algorithms in bioinformatics are not "correct"

Algorithm design (I)

- ➢Exhaustive algorithms (brute force): examine every possible alterative to find the solution
- ➢Branch-and-bound algorithms: omit searching through a large number of alternatives by branch-and-bound or pruning
- ➢Greedy algorithms: find the solution by always choosing the currently ”best” alternative
- ➢Dynamic programming: use the solution of the subproblems of the original problem to construct the solution

Search space



Algorithm design (II)

- ➢Divide-and-conquer algorithms: splits the problem into subproblems and solve the problems independently
- ➢Machine learning: induce models based on previously labeled observations (examples)
- ➢Randomized algorithms: finds the solution based on randomized choices

Pseudo-code

➢Sorting problem: Sort a list of n integers $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$

```
SelectionSort($\boldsymbol{a}$,$n$)
1   for $i \leftarrow 1$ to $n$-1
2       $j \leftarrow$ Index of the smallest element
        among $a_i, a_{i+1}, \ldots, a_n$
3       Swap elements $a_i$ and $a_j$
4   return $\boldsymbol{a}$
```

Example run

$i = 1$:	(**7**,92,87,1,4,3,2,6)
$i = 2$:	(1,**92**,87,7,4,3,2,6)
$i = 3$:	(1,2,**87**,7,4,3,92,6)
$i = 4$:	(1,2,3,**7**,4,87,92,6)
$i = 5$:	(1,2,3,4,**7**,87,92,6)
$i = 6$:	(1,2,3,4,6,**87**,92,7)
$i = 7$:	(1,2,3,4,6,7,**92**,87)
	(1,2,3,4,6,7,87,92)

➢Pseudo-code hides ugly details such as

“Swap elements a_i and a_j”

```
1   $tmp = a_j$
2   $a_j = a_i$
3   $a_i = tmp$
```

or

"$j \leftarrow$ Index of the smallest element among a_i, a_{i+1}, …, a_n"

```
IndexOfMin(array,first,last)
1  $index \leftarrow first$
2  for $k \leftarrow first + 1$ to $last$
3      if $array_k < array_{index}$
4          $index \leftarrow k$
5  return $index$
```

Remember, though, that the devil is in the details!

Recursion

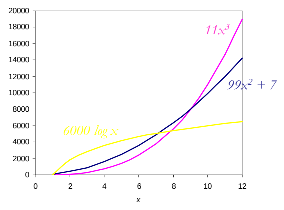
```
RecursiveSelectionSort(a,first,last)
1  if ($first < last$)
2      $index \leftarrow$ Index of the smallest element
       among $a_{first}$, $a_{first+1}$, …, $a_{last}$
3      Swap elements $a_{first}$ and $a_{index}$
4      $a \leftarrow$ RecursiveSelectionSort($a$,$first+1$,$last$)
5  return a
```

Algorithm complexity

- The Big-O notation:
  - the running time of an algorithm as a function of the size of its input
  - worst case estimate
  - asymptotic behavior
- $O(n^2)$ means that the running time of the algorithm on an input of size n is limited by the quadratic function of n

Big-O Notation (I)

- A function $f(x)$ is $O(g(x))$ if there are positive real constants c and x_0 such that $f(x) \leq cg(x)$ for all values of $x \geq x_0$.



Big-O Notation (II)

- A function $f(x)$ is $\Omega(g(x))$ if there are positive real constants c and x_0 such that $f(x) \geq cg(x)$ for all values of $x \geq x_0$
- A function $f(x)$ is $\Theta(g(x))$ if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$
  - g is a tight bound for the function f

Complexity of SelectionSort

- Makes $n - 1$ iterations in the for loop
- Analyzes $n - i + 1$ elements a_i, a_{i+1}, …, a_n in iteration i
- Approximate number of operations:
  - $n + (n-1) + (n-2) + \ldots + 2 + 1 = n(n+1)/2$
  - plus the swapping: $n(n+1)/2 + 3n$
- Thus the algorithm is $O(n^2)$

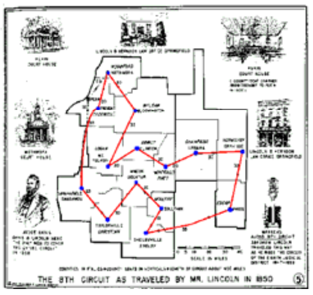
Complexity of RecursiveSelectionSort

- Running time may be described by the recurrence relation:
  - $T(n) = n + T(n-1)$
  - $T(1) = 1$
- Therefore,
  - $T(n) \; = n + T(n-1)$
    $= n + (n-1) + T(n-2)$
    $= n + (n-1) + (n-2) + \ldots + 3 + 2 + T(1)$
    $= O(n^2)$

Tractable versus intractable problems

- Some problems requires polynomial time
  - e.g. sorting a list of integers
  - called tractable problems
- Some problems require exponential time
  - e.g. listing every subset in a list
  - called intractable problems
- Some problems lie in between
  - e.g. the traveling salesman problem
  - called NP-complete problems
  - nobody have proved whether a polynomial time algorithm exists for these problems

Traveling salesman problem



Bioinformatics

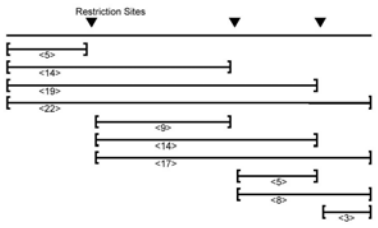
- Sequence analysis and sequence databases
  - First success story: similarity searches to sequence databases e.g. showed the relation between growth proteins and cancer (Doolittle, early 80s)
- Bioinformatics today
  - Functional genomics: determining function for all genes/proteins
  - Systems biology: Predicting whole cell regulations/interactions
  - …
- The ultimate goal: computational simulation of complex living systems

Restriction mapping

- Restriction Enzymes: Discovered in the early 1970's
  - Used as a defense mechanism by bacteria to break down the DNA of attacking viruses.
  - They cut the DNA into small fragments.
- Can also be used to cut the DNA of organisms
  - This allows the DNA sequence to be in a more manageable bite-size pieces

Partial digest example

- Partial digest results in the following 10 restriction fragments

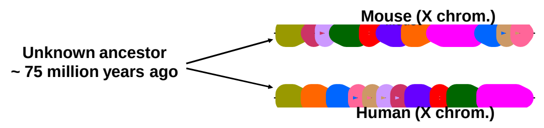


Partial digest problem or restriction mapping

- Goal: Given all pairwise distances between points on a line, reconstruct the positions of those points
- Algorithms: brute force and improvements using branch-and-bound techniques

Genome rearrangements

- The human genome is just the mouse genome cut into about 300 large genomic fragments and then pasted together in a different order



Sorting by reversals

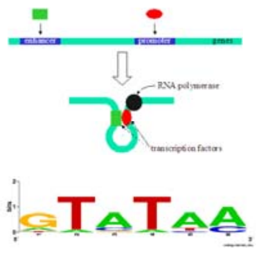
Reversal

1 2 3 4 5 6 $\Rightarrow$ 1 2 -5 -4 -3 6

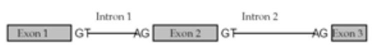
- Goal: Given two permutations, find the shortest series of reversals that transform one permutation into the other
- Algorithms: Greedy search and approximation algorithms

Motif finding

- Transcription factors regulate specific genes by binding selectively to sequence motifs
- Motif Finding Problem: Given a list of sequences, find the "best" pattern that appears in all of the sequences
- Algorithms: exhaustive, greedy and randomized strategies



Gene prediction



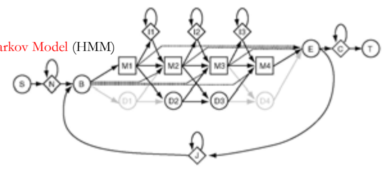
- Gene prediction: Locate genes in a genomic sequence
- Statistical: coding segments (exons) have typical sequences on either end and use different subwords than non-coding segments (introns)
- Similarity-based: many human genes are similar to genes in mice, chicken, or even bacteria. Therefore, already known mouse, chicken, and bacterial genes may help to find human genes (comparative genomics)
- Algorithm: Dynamic programming

Multiple sequence alignment modeling

```
FOS_RAT        PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE      PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK      SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE     PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN     PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
```

Model a multiple alignment of e.g. a protein family and use the model to recognize other family members (Pfam)

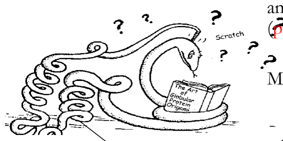
Hidden Markov Model (HMM)



Structure prediction

Goal: discover nature's algorithm for specifying the three–dimensional structure of proteins from their amino acid sequences (protein folding problem)

Method: HMMs

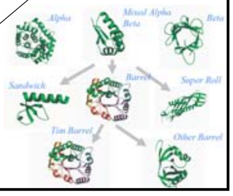


```
KKAVINGEQIRSISDLHQTLKKELALPEYYGENLDALWDCL
TGWVEYPLVLEWRQFEQSKQLTENGAESVLQVFREAKAEGC
DITIILS

KHCTISGRAVHSLDELYDEIARQLPLPDYFGRNLDALWDVL
STDIEGPVELIWEDSEHSKRSMGKDYERVVALLKDLTEERE
DFRIV

IIGSKIYTEQDFHNQISKIFSIQDYYGNNLDALWDLLSTNV
ERPITLVWKDAMFSKNQLENIFIEIVNVLERVKKQDED

QSKQEVLETIATSFLFPKHFGKNYDALYDCLTDLVQFVIVL
E--QLPVAQKFDKEGRETLLDVFREA
```


Alpha
Mixed Alpha Beta
Beta
Sandwich
Barrel
Super Roll
Tim Barrel
Other Barrel