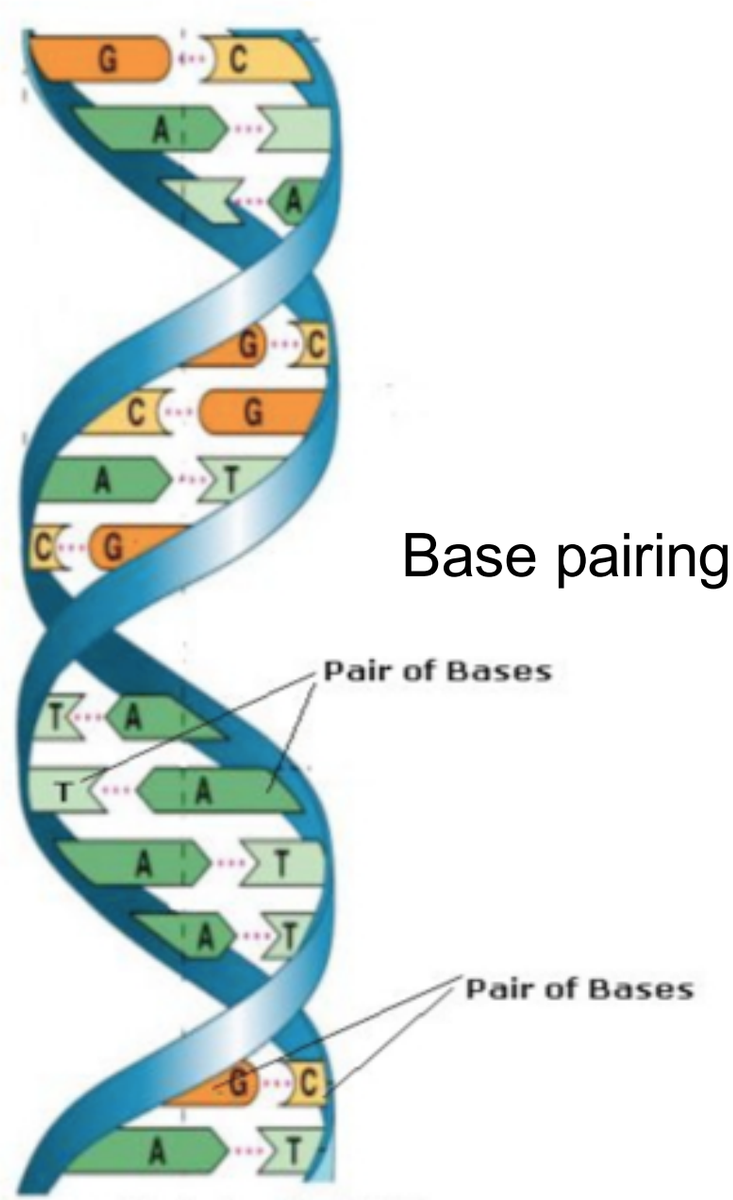
CS 466
Introduction to Bioinformatics

Instructor: Jian Peng

Biological sequences

DNA



Base Pairing in DNA Double Helix

Base pairing property

=

The DNA Molecule

5’

G -- C
A -- T
T -- A
G -- C
C -- G
G -- C
T -- A
G -- C
T -- A
T -- A
A -- T
A -- T
C -- G
T -- A

3’

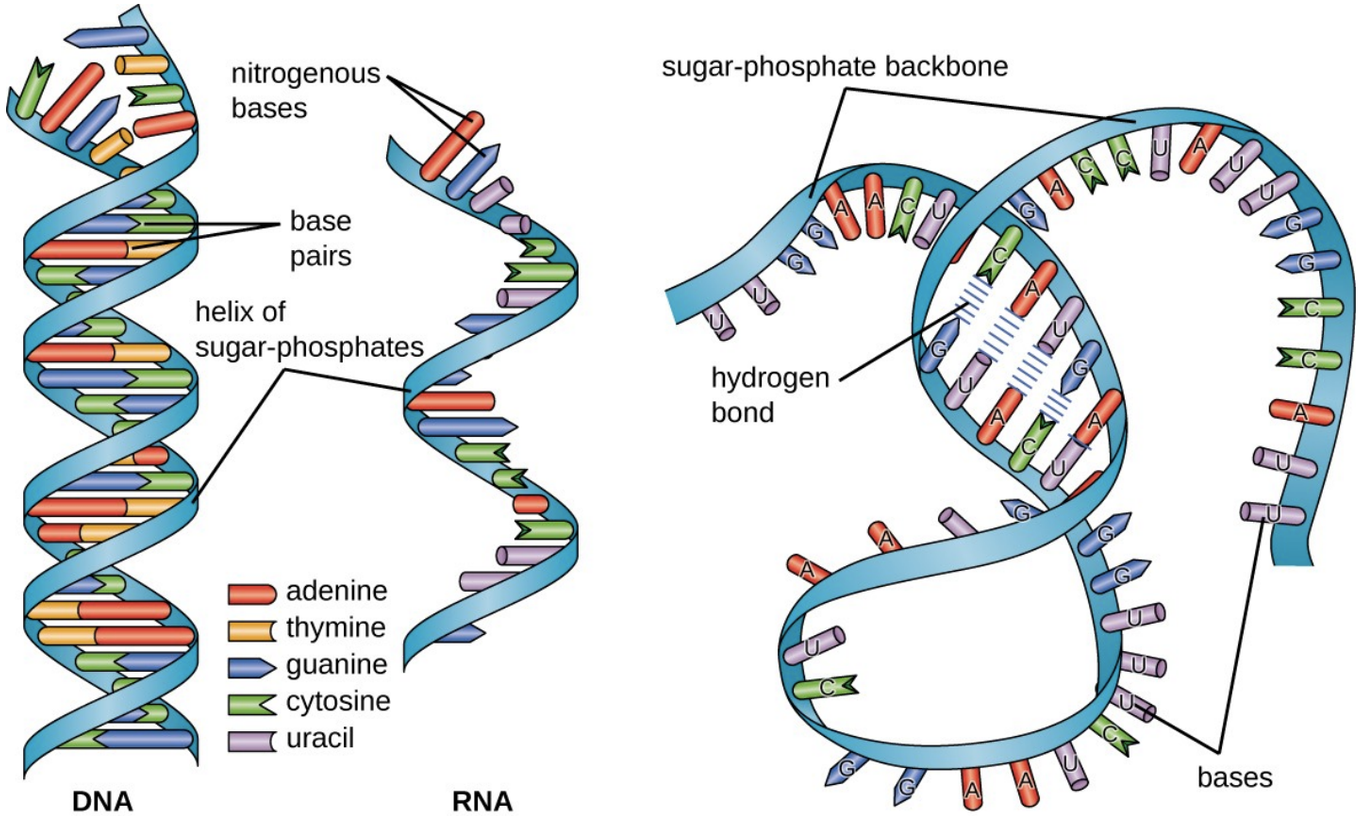
DNA to chromosome

SOURCE: http://www.microbe.org/espanol/news/human_genome.asp

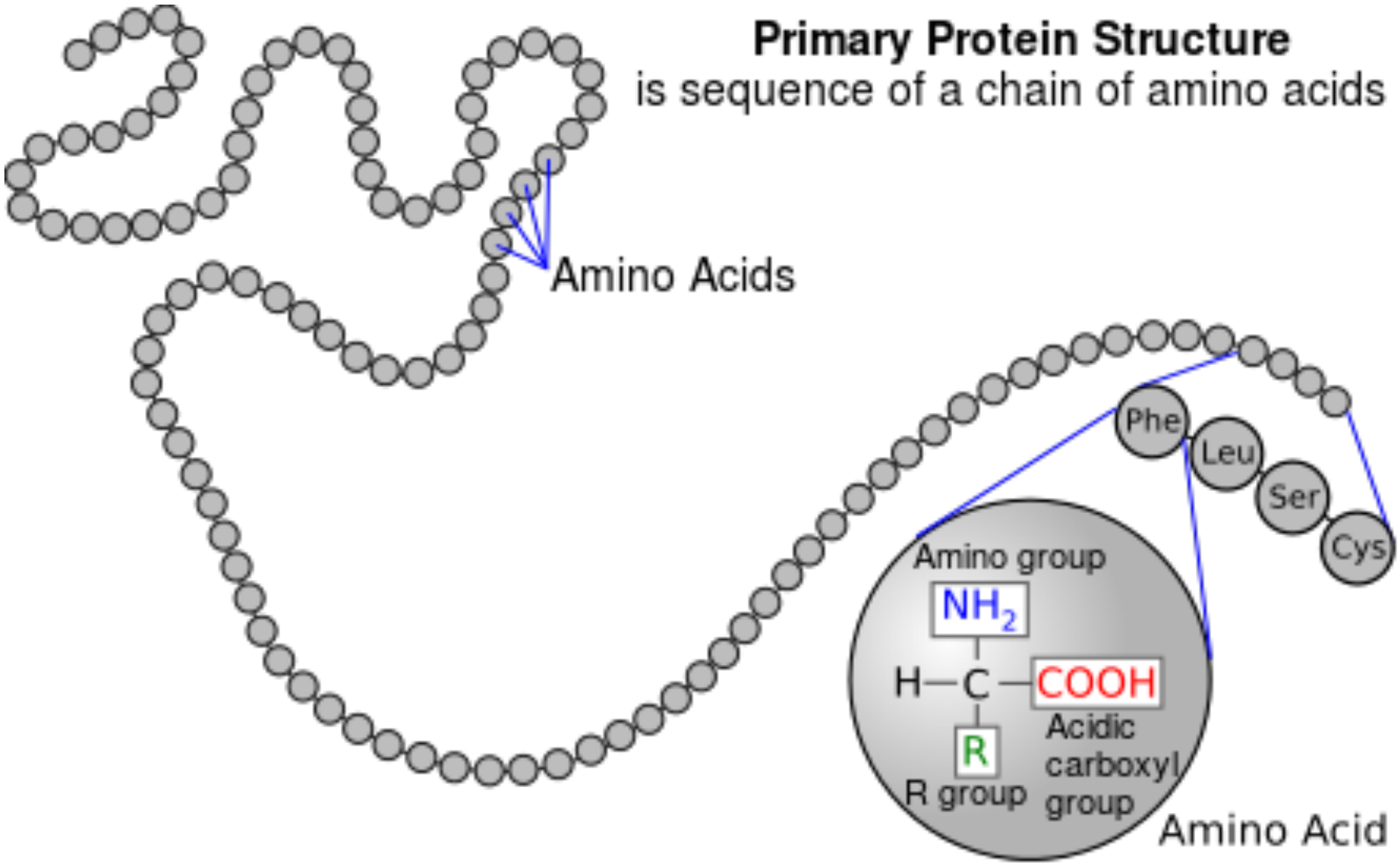
What is RNA?

RNA = ribonucleic acid

- “U” instead of “T”
- Usually single stranded
- Has base-pairing capability
  - Can form simple non-linear structures
- Life may have started with RNA



Protein sequence



A short summary: string transformation

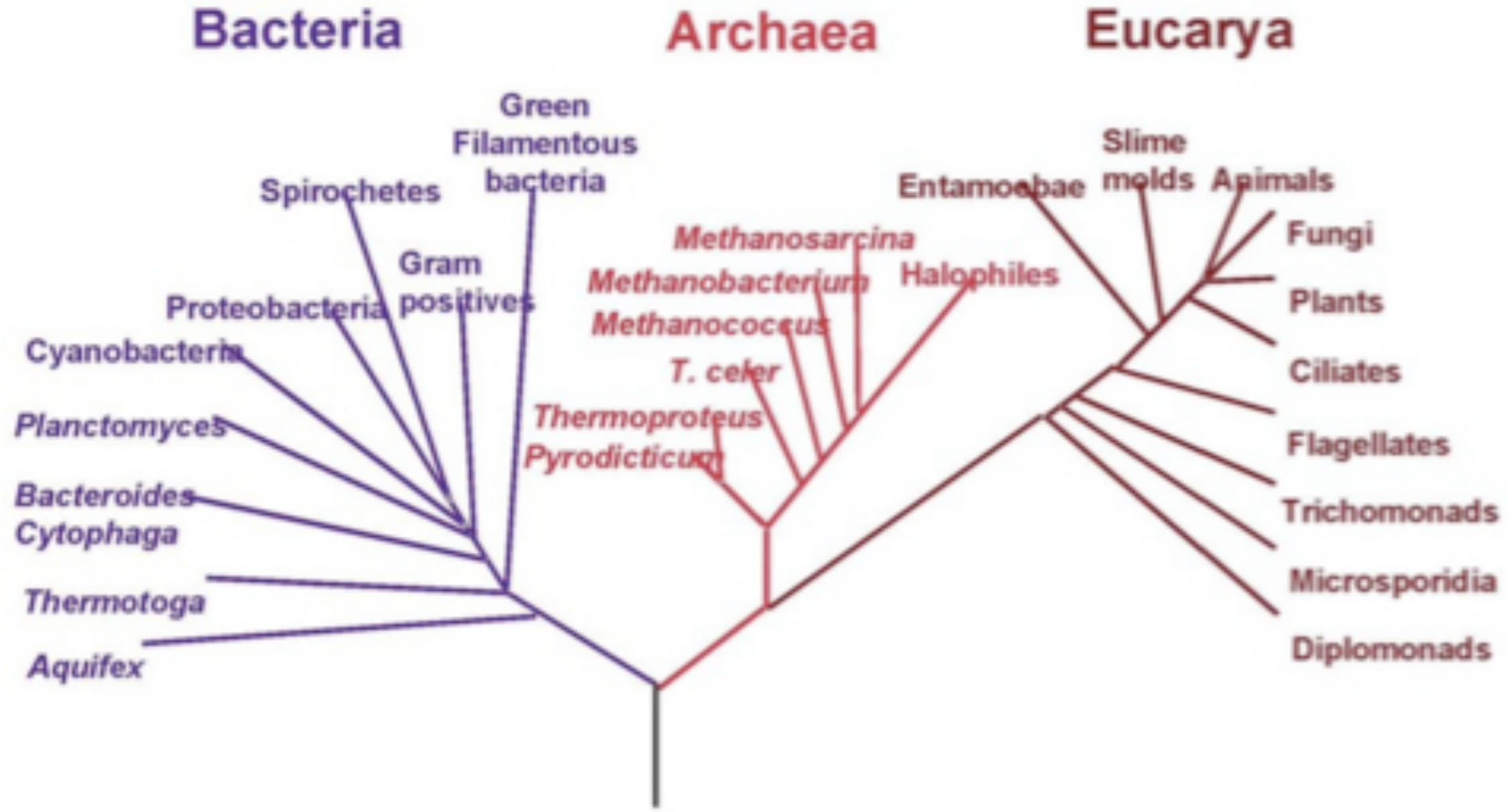
- DNA = nucleotide sequence
  - Alphabet size = 4 (A,C,G,T)

- DNA to mRNA (single stranded)
  - Alphabet size = 4 (A,C,G,U)

- mRNA to amino acid sequence
  - Alphabet size = 20

- Amino acid sequence “folds” into 3-dimensional protein

Phylogenetic Tree of Life



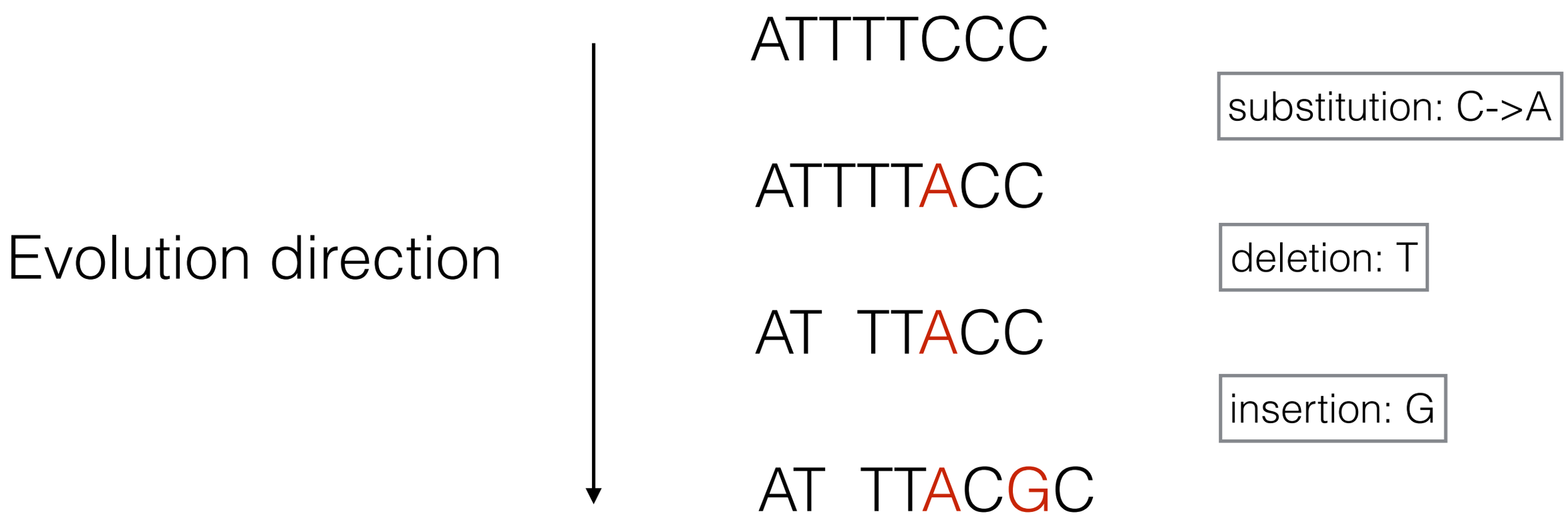
determined by DNA sequences

Evolution theory

- All organisms share the genetic code
- Similar genes across species
- Probably had a common ancestor
- Genomes are a wonderful resource to trace back the history of life

Evolutionary process of sequences

- Substitutions
- Insertions and Deletions
- Representing an alignment; “gaps”



Sequence alignment

Correspondence between bases of two DNA sequences, or between amino acids of two protein sequences

Alignment : 2 x k matrix (k ≥ m, n)

V = ACCTGGTAAA n = 10

W = ACTGCGTATA m = 10

8 matches
1 mismatches
1 deletions
1 insertions

V	A	C	C	T	G	—	G	T	A	A	A
W	A	C	—	T	G	C	G	T	A	T	A

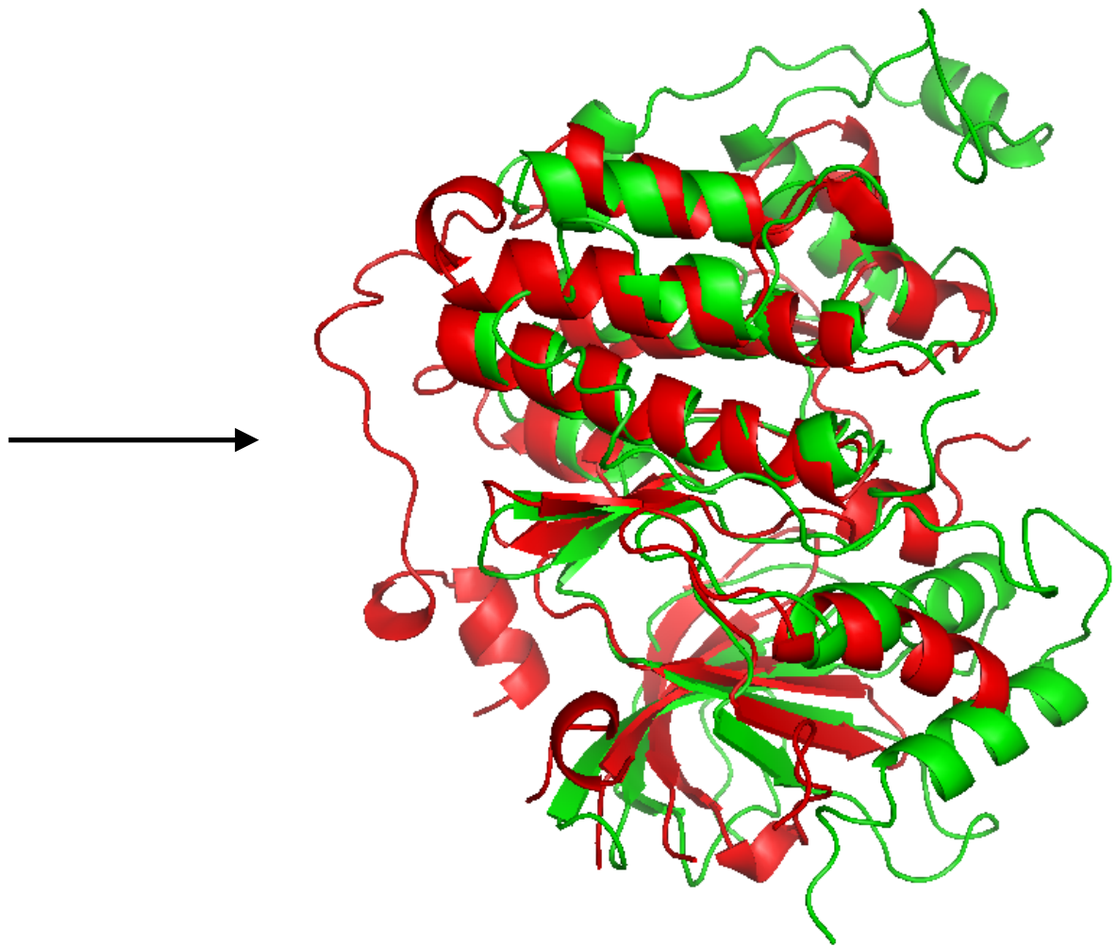
Applications of sequence alignment

- Find similarity between two DNA sequences that have evolved from a common ancestor
- Estimate evolutionary "distance" (time) between two related sequences, e.g., phylogeny reconstruction
- Enables inferences about evolutionary process, e.g., A map of recent positive selection in the Human Genome – Voight et al, PLoS Bio, 2006.

Applications of sequence alignment

- Sequence alignments of RNA or amino acid sequences help in structure prediction

```
Query pemvrgqvfdVGPRYTNLSYIGEGAYGMVCSAYDNVNKVRVAIKKIspfehqtyCQRTLR    60
ident                 | ||  | |         |  ||||
Sbjct ---qallrilKETEFKKIKVLGSGAFGTVYKGLWIPVKIPVAIKEL------anKEILDE    51

Query eIKILLRFRHENIIGINDIIRAptieqmKDVYIVQDLM-ETDLYKLLKTQH--LSNDHIC  117
ident                  |          |     ||    |
Sbjct -AYVMASVDNPHVCRLLGICLT------STVQLITQLMpFGCLLDYVREHKdnIGSQYLL  104

Query YFLYQILRGLKYIHSANVLHRDLKPSNLLLNTTCDLKICD-FGLAR---vadVATRWYRA  173
ident    ||  |  |       ||||   | |  |    || |                |
Sbjct NWCVQIAKGMNYLEDRRLVHRDLAARNVLVKTPQHVKITDfGLAKLlgaeekKVPIKWMA  164

Query PEIMLNsKGYTKSIDIWSVGCILAEMLSN-RPIFPgKHYLDQLNHILGILgspsqedlnc  232
ident  |  |    ||   | || |    |
Sbjct LESILH-RIYTHQSDVWSYGVTVWELMTFgSKPYDgIPASEISSILEKGE----------  213

Query iinlkarnyllslphknkvpwnrLFPNADSKALDLLDKMLTFNPHKRIEVEQALAHpYLE  292
ident                        |          |         |
Sbjct --------------------rlpQPPICTIDVYMIMVKCWMIDADSRPKFRELIIE-FSK  252

Query QYydpsdepiaeapfkfdmelddlpkeklkelifeetarfqpg  335
ident
Sbjct MA--rdpqrylviqgdermhlpsptdsnfyralmdvvdadeyl  293
```



How to compute an alignment?

Alignment : 2 x k matrix (k ≥ m, n)

V = ACCTGGTAAA n = 10

W = ACTGCGTATA m = 10

8 matches
1 mismatches
1 deletions
1 insertions

V	A	C	C	T	G	—	G	T	A	A	A
W	A	C	—	T	G	C	G	T	A	T	A

“Goodness” of alignments

Given two sequences, there are many possible alignments

```
ATTTTCCC
ATTTACGC
```

```
ATTT-TCCC
ATTTA-CGC
```

```
ATTTTCCC--------
--------ATTTACGC
```

“Goodness” of alignments

Given two sequences, there are many possible alignments

```
ATTTTCCC

ATTTACGC
```

```
ATTT-TCCC

ATTTA-CGC
```

```
ATTTTCCC--------

--------ATTTACGC
```

Edit distance: the total number of substitutions, insertions and deletions needed to transform one sequence to another

“Goodness” of alignments

Given two sequences, there are many possible alignments

```
ATTTTCCC
ATTTACGC
```

distance=2

```
ATTT-TCCC
ATTTA-CGC
```

distance=3

```
ATTTTCCC--------
--------ATTTACGC
```

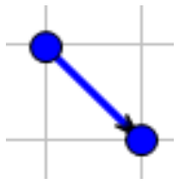
distance=16

Edit distance: the total number of substitutions, insertions and deletions needed to transform one sequence to another

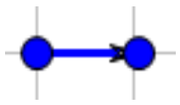
How to compute the best alignment?

Enumeration of all possible alignments

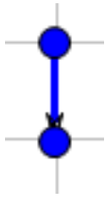
Match

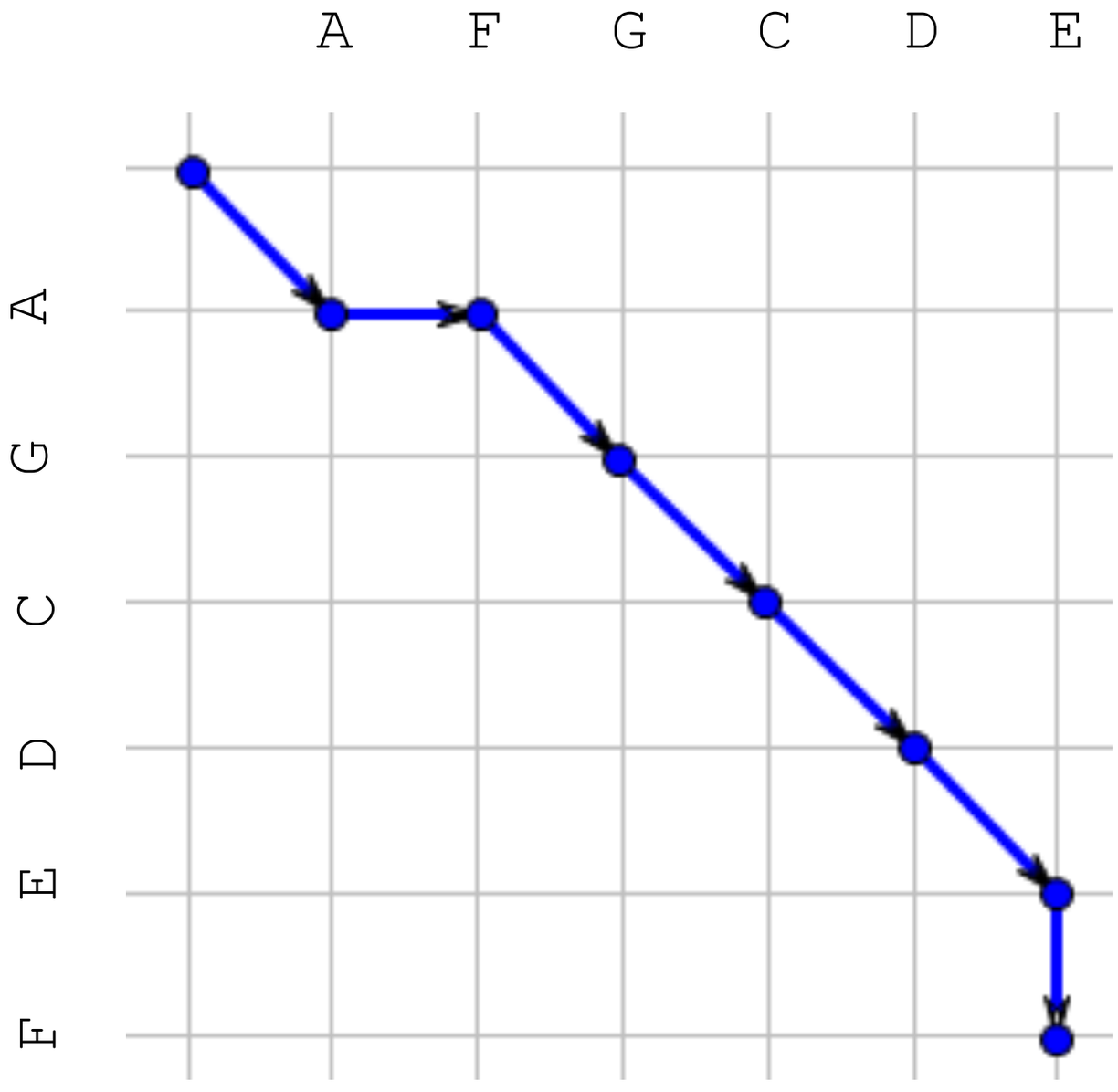


Insertion_X



Insertion_Y





A-GCDEF
AFGCDE-

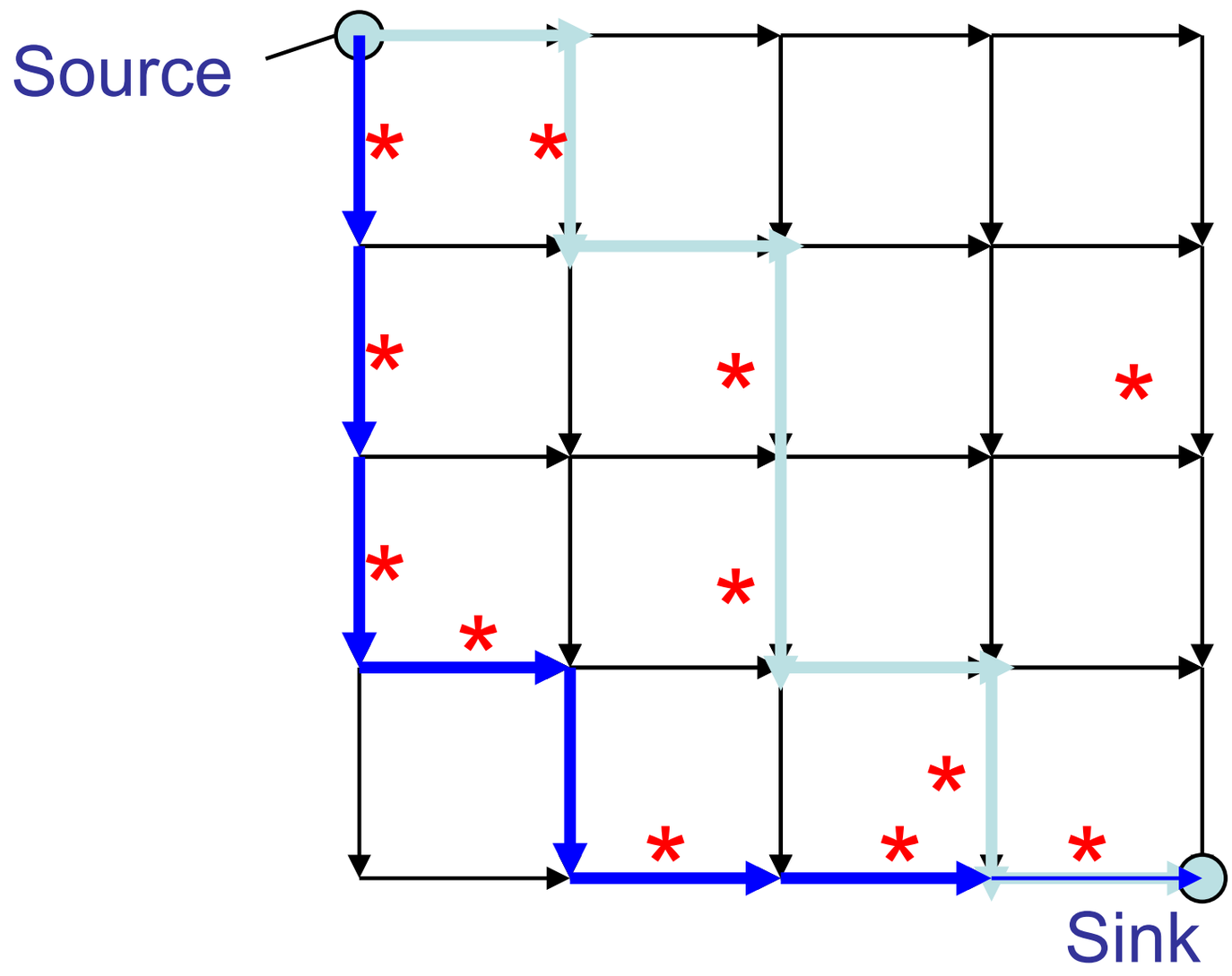
Very expensive

Introduction to dynamic programming

A simplified example

Manhattan tourist problem

Imagine seeking a path (from source to sink) to travel (only eastward and southward) with the most number of attractions (*) in the Manhattan grid



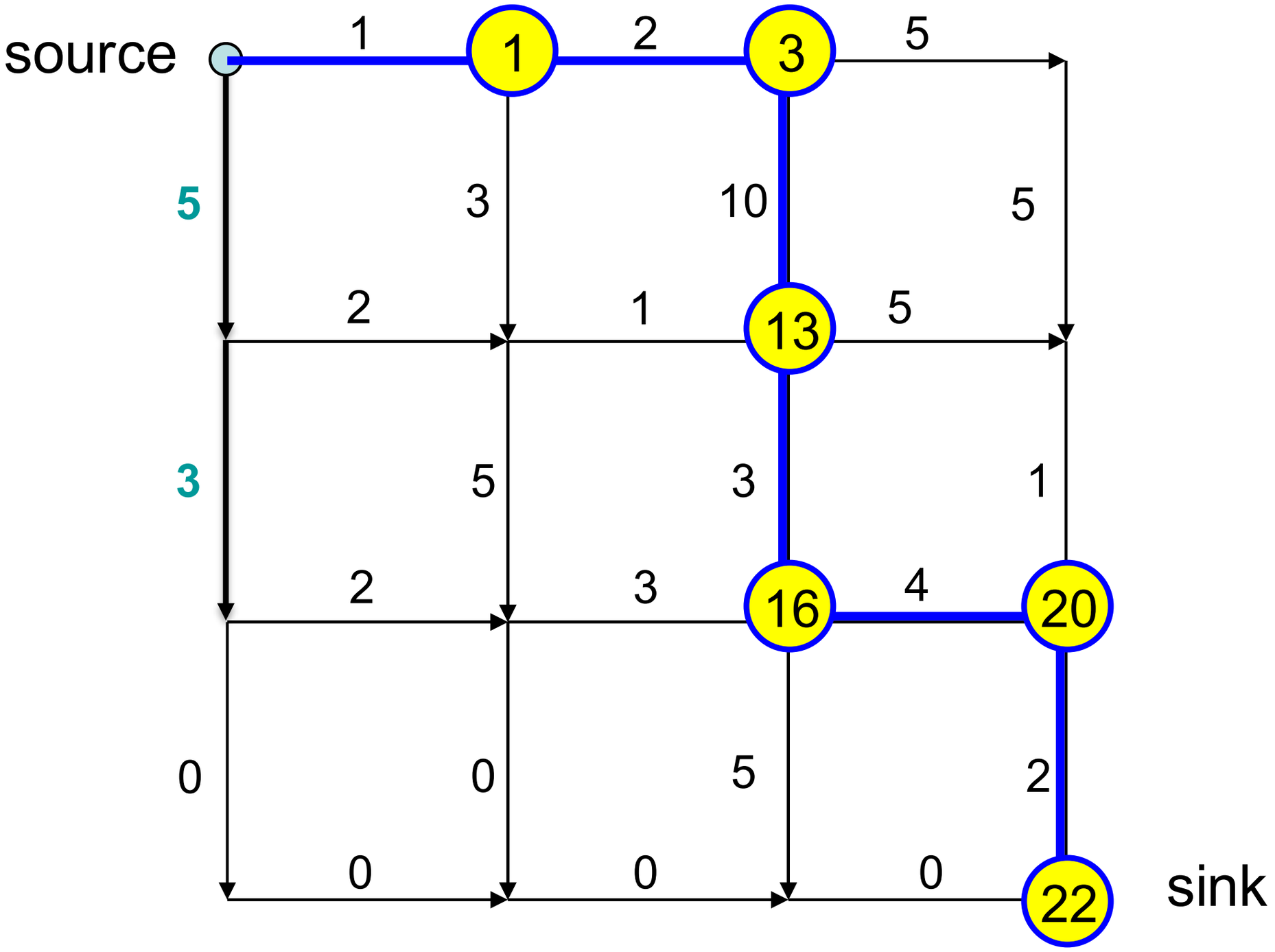
Problem formulation

Goal: Find a “most weighted” path in a weighted grid. (Weights may not be just 0/1.)

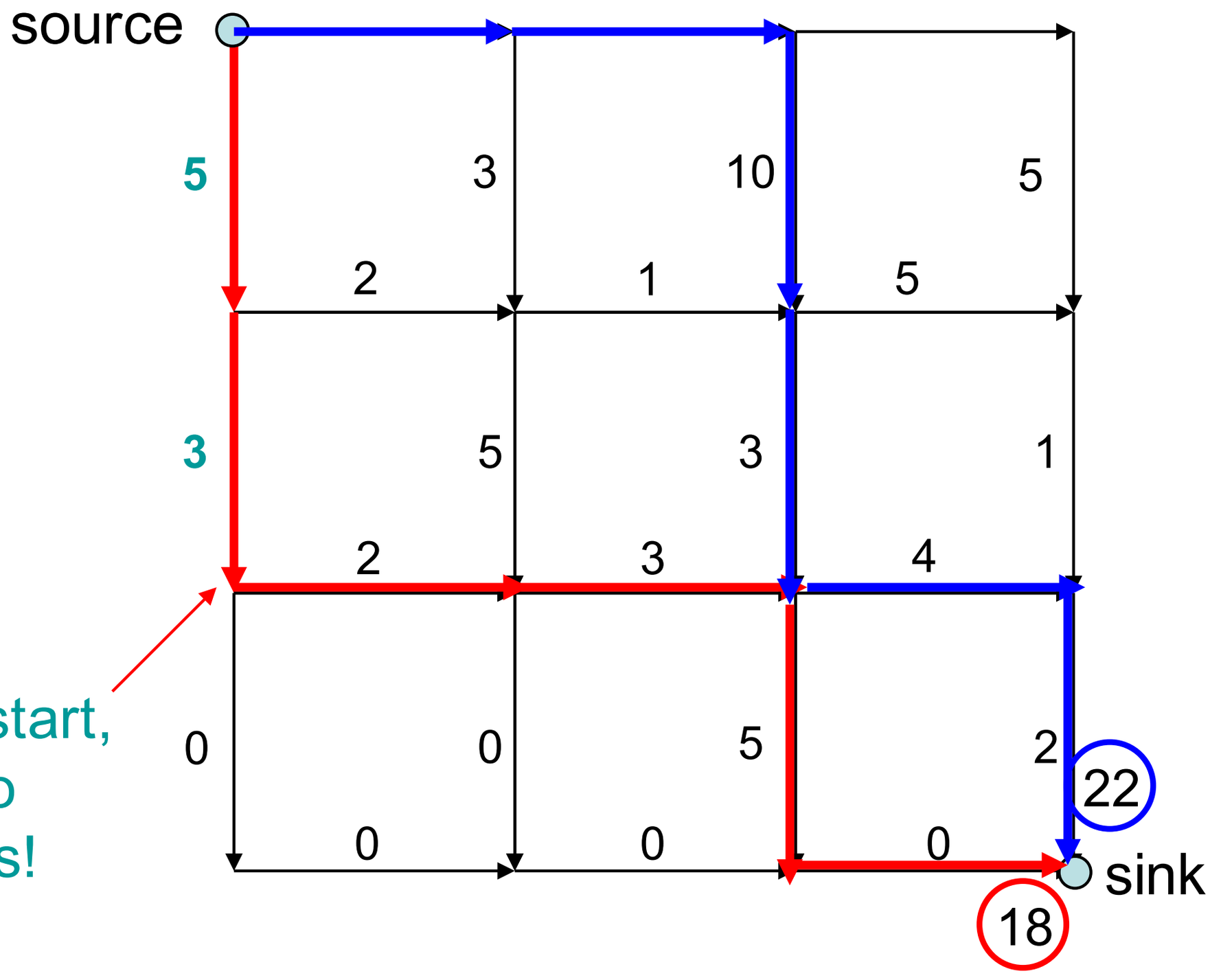
Input: A weighted grid **G** with two distinct vertices, one labeled “source” and the other labeled “sink”

Output: A “most weighted” path in **G** from “source” to “sink”

An example



Would a greedy algorithm work?



promising start,
but leads to
bad choices!

How about a recursive solution?

Function **MT**(*n,m*)

1. *x = MT(n-1,m)+*
   weight of the edge from *(n-1,m) to (n,m)*
2. *y = MT(n,m-1)+*
   weight of the edge from *(n,m-1) to (n,m)*
3. **return** *max{x,y}*

MT(x, y) returns the “most weighted” path from point (x, y) to the “sink”.

Why this is not efficient?

- MT(n,m) needs MT(n, m-1) and MT(n-1, m)
- Both of these need MT(n-1, m-1)
- So MT(n-1, m-1) will be computed at least twice
- Dynamic programming: the same idea as this recursive algorithm, but keep all intermediate results in a table and reuse