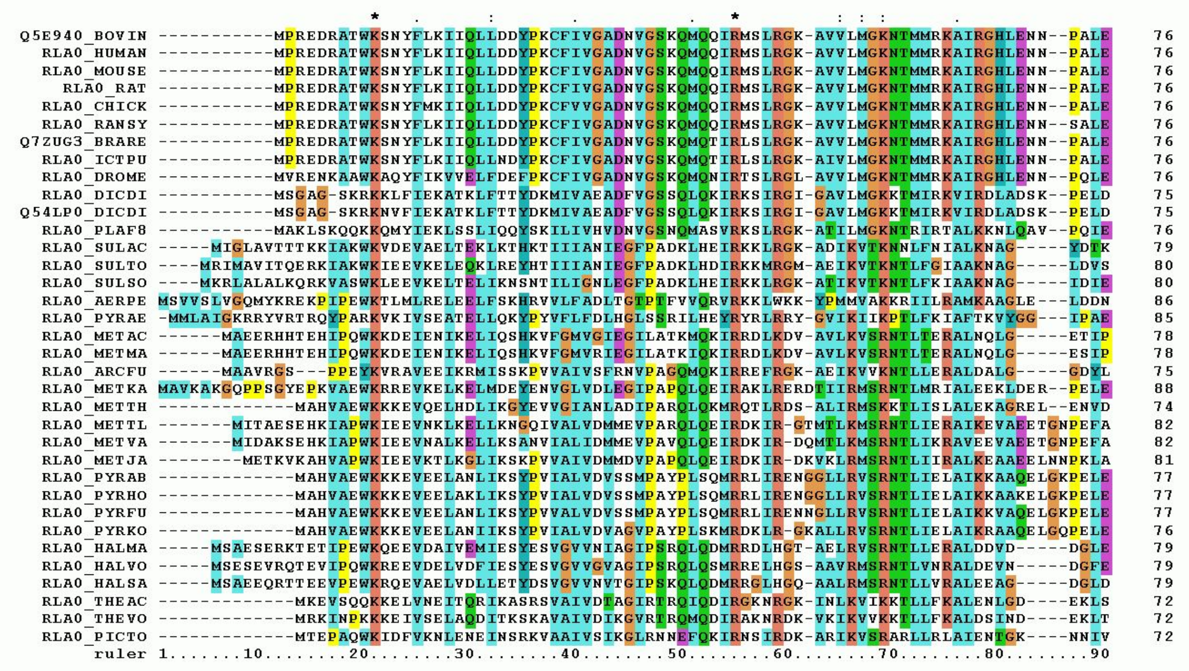
Comp 555 - BioAlgorithms - Spring 2020



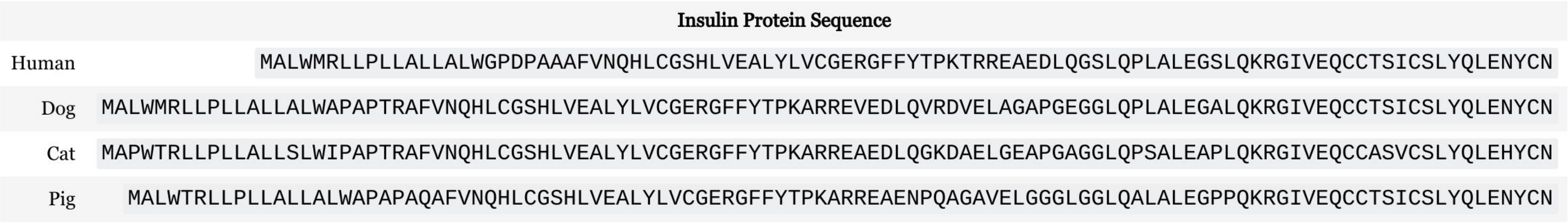
```
                                *   .      :       .          .   *         : : :      .
Q5E940_BOVIN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_HUMAN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_MOUSE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
    RLA0_RAT -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_CHICK -----------MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_RANSY -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--SALE    76
Q7ZUG3_BRARE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_ICTPU -----------MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKQMQTIRLSLRGK-AIVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_DROME -----------MVRENKAAWKAQYFIKVVELFDEFPKCFIVGADNVGSKQMQNIRTSLRGL-AVVLMGKNTMMRKAIRGHLENN--PQLE    76
  RLA0_DICDI -----------MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
Q54LP0_DICDI -----------MSGAG-SKRKNVFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
  RLA0_PLAF8 -----------MAKLSKQQKKQMYIEKLSSLIQQYSKILIVHVDNVGSNQMASVRKSLRGK-ATILMGKNTRIRTALKKNLQAV--PQIE    76
  RLA0_SULAC -----MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAG-----YDTK    79
  RLA0_SULTO ----MRIMAVITQERKIAKWKIEEVKELEQKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAG-----LDVS    80
  RLA0_SULSO ----MKRLALALKQRKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAG-----IDIE    80
  RLA0_AERPE MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLE---LDDN    86
  RLA0_PYRAE -MMLAIGKRRYVRTRQYPARKVKIVSEATELLQKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGG---IPAE    85
  RLA0_METAC ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVGIEGILATKMQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ETIP    78
  RLA0_METMA ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ESIP    78
  RLA0_ARCFU ------MAAVRGS---PPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGQMQKIRREFRGK-AEIKVVKNTLLERALDALG-----GDYL    75
  RLA0_METKA MAVKAKGQPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPQLQEIRAKLRERDTIIRMSRNTLMRIALEEKLDER--PELE    88
  RLA0_METTH -------------MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGREL--ENVD    74
  RLA0_METTL -------MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPARQLQEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA    82
  RLA0_METVA -------MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA    82
  RLA0_METJA --------METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPQLQEIRDKIR-DKVKLRMSRNTLIIRALKEAAEELNNPKLA    81
  RLA0_PYRAB -------------MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE    77
  RLA0_PYRHO -------------MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE    77
  RLA0_PYRFU -------------MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE    77
  RLA0_PYRKO -------------MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE    76
  RLA0_HALMA -----MSAESERKTETIPEWKQEEVDAIVEMIESYESVGVVNIAGIPSRQLQDMRRDLHGT-AELRVSRNTLLERALDDVD-----DGLE    79
  RLA0_HALVO -----MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVN-----DGFE    79
  RLA0_HALSA -----MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAG-----DGLD    79
  RLA0_THEAC -------------MKEVSQQKKELVNEITQRIKASRSVAIVDTAGIRTRQIQDIRGKNRGK-INLKVIKKTLLFKALENLGD----EKLS    72
  RLA0_THEVO -------------MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTRQMQDIRAKNRDK-VKIKVVKKTLLFKALDSIND----EKLT    72
  RLA0_PICTO -------------MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFQKIRNSIRDK-ARIKVSRARLLRLAIENTGK----NNIV    72
       ruler 1.......10........20........30........40........50........60........70........80........90
```

- Problem set #3 is due next Tuesday
- Midterm is set for next Thursday, and covers up to the previous lecture.
- Open notes, open internet sans messaging apps.
- Jupyter Notebook

Comparing Sequences

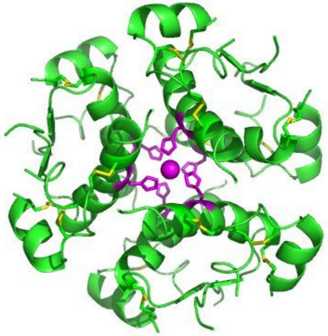
Sequence Similarity

- A common problem in biology



	Insulin Protein Sequence
Human	MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
Dog	MALWMRLLPLLALLALWAPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEDLQVRDVELAGAPGEGGLQPLALEGALQKRGIVEQCCTSICSLYQLENYCN
Cat	MAPWTRLLPLLALLSLWIPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEDLQGKDAELGEAPGAGGLQPSALEAPLQKRGIVEQCCASVCSLYQLEHYCN
Pig	MALWTRLLPLLALLALWAPAPAQAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAENPQAGAVELGGGLGGLQALALEGPPQKRGIVEQCCTSICSLYQLENYCN

- All similar, but how similar?
- How do you measure similarity?
- Does Hamming distance work here?
- Uses
  - To establish a *phylogeny*
  - To identify *functional* or *conserved* components of the sequence



Hand Alignments

- Not that long ago, many alignments were done by hand

```
Human : MALWMRLLPLLALLALWGPdPAaAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQ_____________GSLQPLALEGs_LQKRGIVEQCCTSICSLYQLENYCN
        ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||             |||||||||||||||||||||||||||||||||||||
  Dog : MALWMRLLPLLALLALWAPAPtRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREvEDLQvrDVELaG_APGeGGLQPLALEGA_LQKRGIVEQCCTSICSLYQLENYCN
        ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  |||| | ||| ||||||||| | |||||||||||||||||||||||||
  Cat : MApWtRLLPLLALLsLWiPAPtRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEDLQgkDaEL_GeAPGaGGLQPsALE_APLQKRGIVEQCCaSvCSLYQLEHYCN
        ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  |||| | ||  |||||||||||| ||||||||||||||||||||||||
  Pig : MALWtRLLPLLALLAlWAPAPAqAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEnpQagaVEL_Gggl__GGLQaLALEGpP_QKRGIVEQCCTSICSLYQLENYCN
                               AFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAE                             QKRGIVEQCC SICSLYQLENYCN
```

- Long conserved regions are shown below
- Solution strategy?
- Is this a well defined problem?
  - Is there an optimal or best solution?
  - Did we find it?
- By the way, this is an easy case. Within vertebrates, the amino acid sequence of insulin is strongly conserved.

The Alignment Game

Let's consider only 2 sequences, and estabish “alignment” rules as if it were a game.

- Rules:
    - You must remove all characters from both sequences
    - There are 3 possible moves at any point in the game.
    - Each move removes at least one character from one of the two given strings
    - Pressing [Match] removes one left-most character from both sequences
        - You get 1 point if the characters match, otherwise you get 0 points
    - Pressing [Del] removes the left-most character from the top sequence
        - You lose 1 point
    - Pressing [Ins] removes the left-most character from the bottom sequence
        - You lose 1 point
    - Your point total is allowed to go negative
- Objective: Get the most points

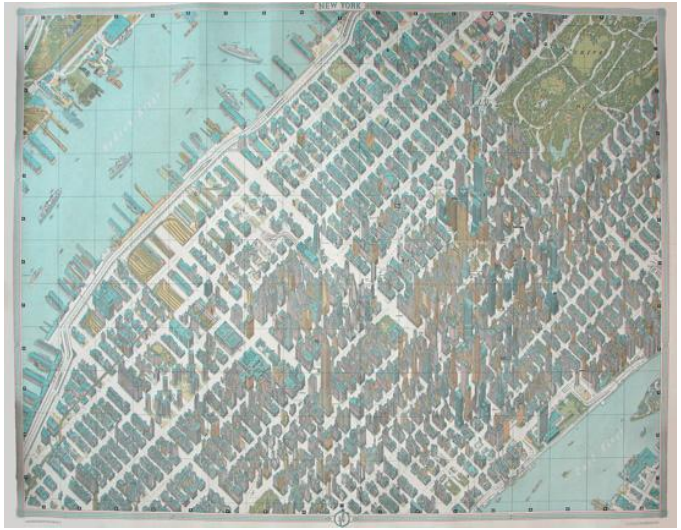
How do you get the highest possible score?

- The solution may not be unique
- How many presses?
  - Minimum moves = *Max(len(top), len(bot))*
  - Maximum moves = *len(top) + len(bot)*
- How many possible moves?
  - Less than $3^{\text{len(top) + len(bot)}}$
- How big for our problem instance?
  - len(Human) = 98, len(dog) = 110
  - $3^{208} \approx 1.73 \times 10^{90}$, almost a googol (not a google)
- What algorithm solves this problem?
  - Make each move by considering only a short horizon following the current aligment thus far

There is an effcient solution

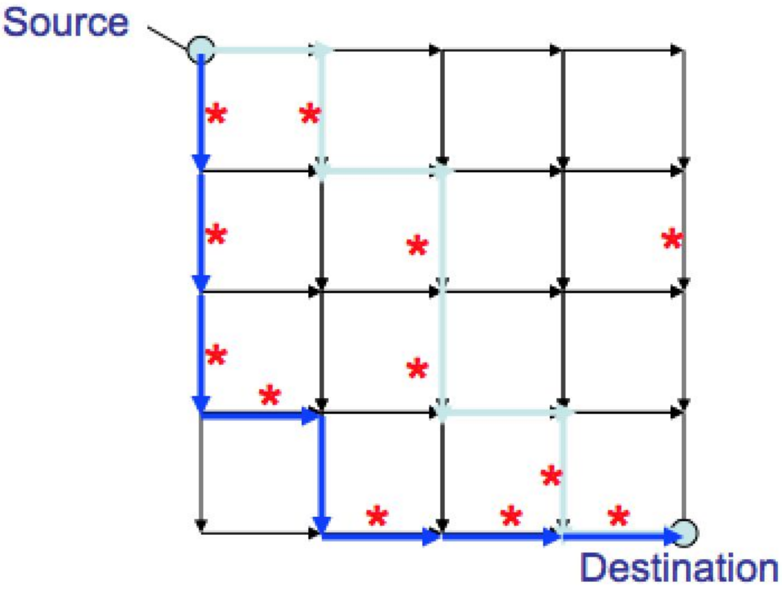
- It relies on a rather suprising idea
- The best score can be found for the len(top) and len(bot) strings by finding the best score for every pair of substrings len(top[0:n]) and len(bot[0:m]) for all values of n up to len(top) and m up to len(bot)
- Finding this solution requires only O(len(top)len(bot)) steps
- It also requires a table of size Max(len(top),len(bot))
- But before we solve this problem, let's look at another related related problem

Finding a best city tour on a Manhattan grid



Manhattan Tourist Problem (MTP)

Imagine seeking a path from a given source to given destination in a Manhattan-like city grid that maximizes the number of attractions (*) passed. With the following caveat– at every step you must make progress towards the goal. We treat the city map as a graph, with a *vertices* at each intersection, and *weighted edges* along each block. The weights are the number of attractions along each block.



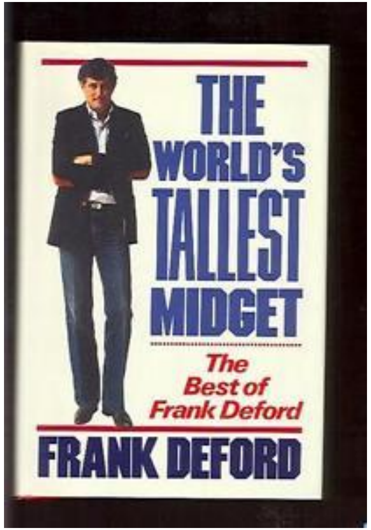
Manhattan Tourist Game

Goal: Find the maximum weighted shortest path in a grid.

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

Output: A *shortest* path in G from *source* to *destination* with the *greatest* weight

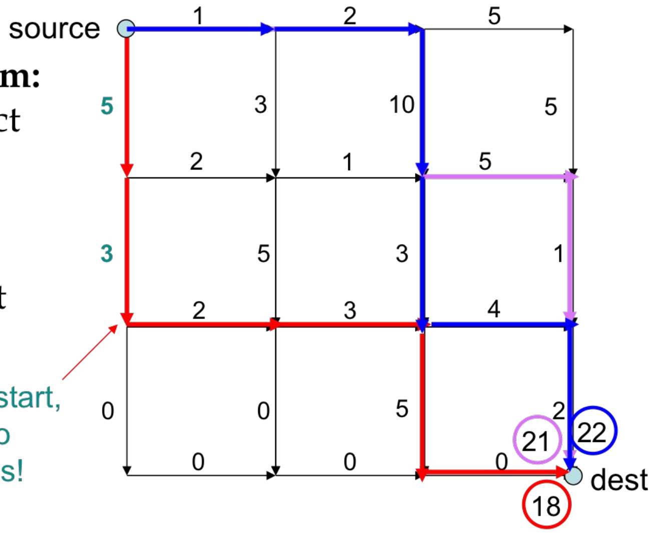
- There are many *shortest* paths that go south 4 blocks and east 4 blocks
- Of those paths, which sees the most sites?



MTP: A Greedy Algorithm Is Not Optimal

Greedy Algorithm:
At each step select the maximum weight block.

Greed has a short horizon



promising start, but leads to bad choices!

-- Short horizon greedy

-- Long horizon greedy

-- Better, but is it optimal?

Different types of ***Greedy***

- *Short horizon*: At each block select the direction where the next block offers the most attractions
- *Long horizon*: Look ahead at all streets between your current position and the destination, and then go down streets with the most attractions

A New Solution Strategy

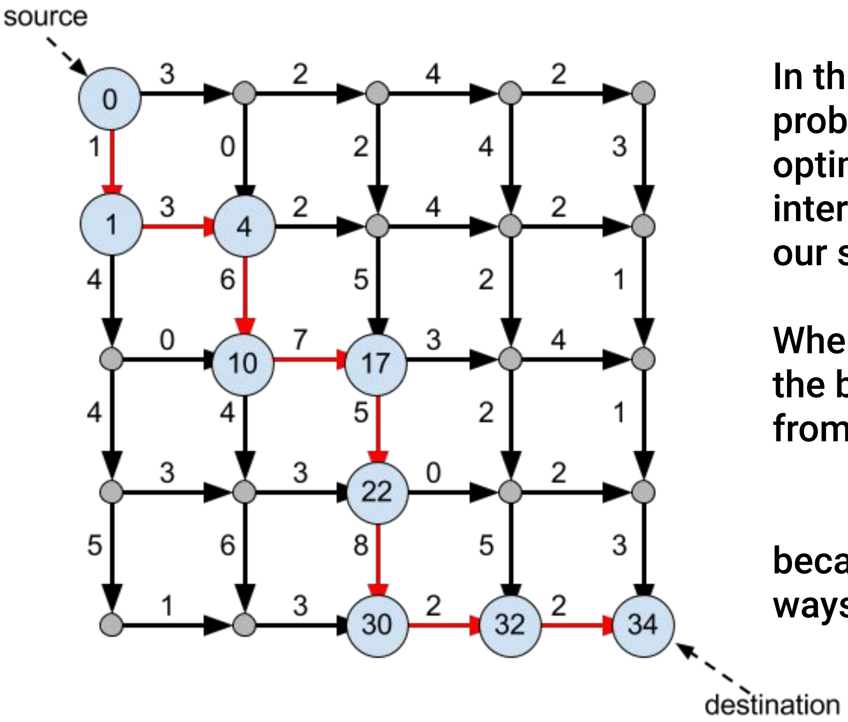
Dynamic Programming is a technique for *computing recurrence relations efficiently by storing and reusing intermediate results*

Three keys to constructing a dynamic programming solution:

1. Formulate the answer as a recurrence relation
2. Consider all instances of the recurrence at each step
   (In our case this means all paths that lead to a vertex or intersection).
3. *Order evaluations so you will always have precomputed any needed partial results*

Irony: Often the most effcient approach to solving a specific problem
involves solving ***every*** smaller subproblem.

MTP Dynamic Program Solution



In this case our smaller sub problems are finding the optimal path to "every" intersection that lies between our source and destination.

Where the optimal path to (i,j) is the better of the optimal paths from:

(i-1,j) to (i,j) or

(i,j-1) to (i,j),

because those are the only ways to get to (i,j)

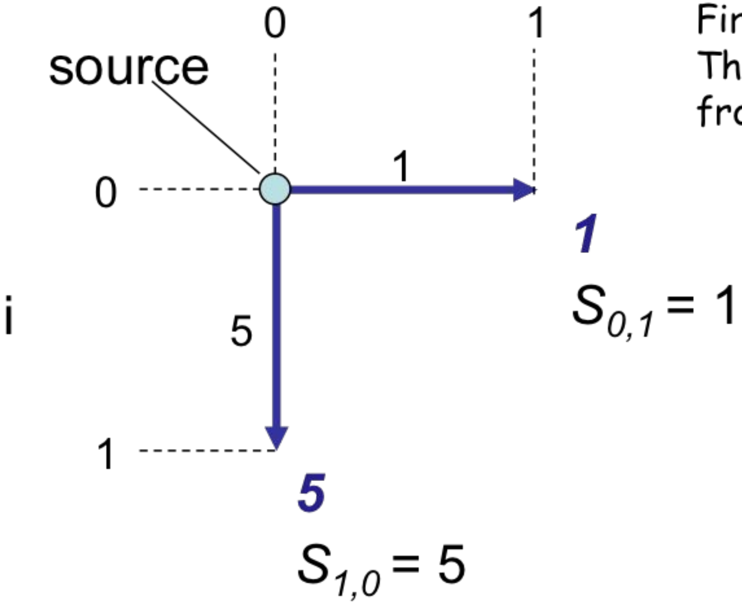
The solution may not be unique, but it will have the best possible, optimal, score

MTP Dynamic Program Strategy

- Instead of solving the Manhattan Tourist problem directly, (i.e. the path from (0,0) to (n,m)) we will solve a more general problem: find the longest path from (0,0) to any arbitrary vertex (i,j).
- If the longest path from (0,0) to (n,m) passes through some vertex (i,j), then the path from (0,0) to (i,j) must be the longest. Otherwise, you could increase the weight along your path by changing it.

MTP: Dynamic Program

- Calculate optimal path score for *every* vertex in the graph between our source and destination
- Each vertex's score is the maximum of the prior vertices score plus the weight of the connecting edge in between



$S_{0,1} = 1$

$S_{1,0} = 5$

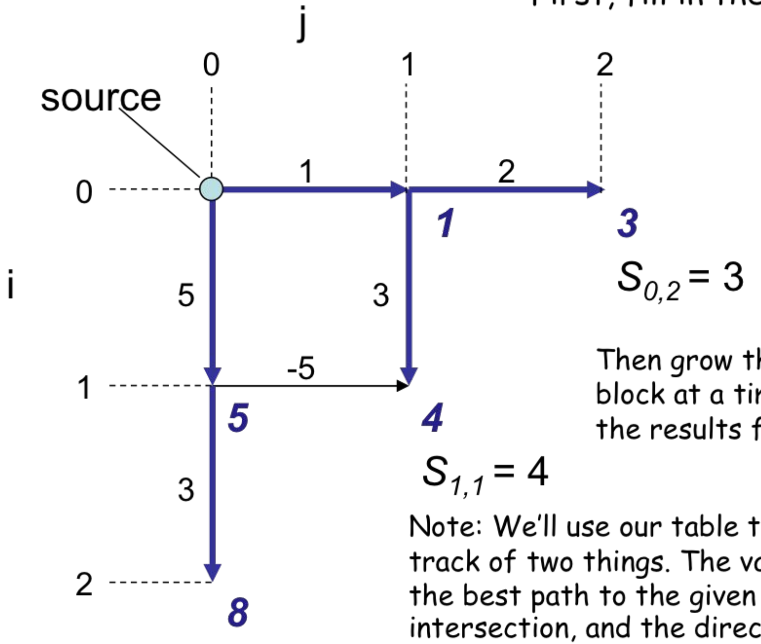
First, fill in the easy ones!
Those 1 block
from the source

MTP: Dynamic Program Continued

Consider all destinations 2 blocks from the source.

Notice I have allowed "negative" edge weights… assume these are the number of things that your guide book suggests you should avoid at all cost!

First, fill in the easy ones!



$S_{0,2} = 3$

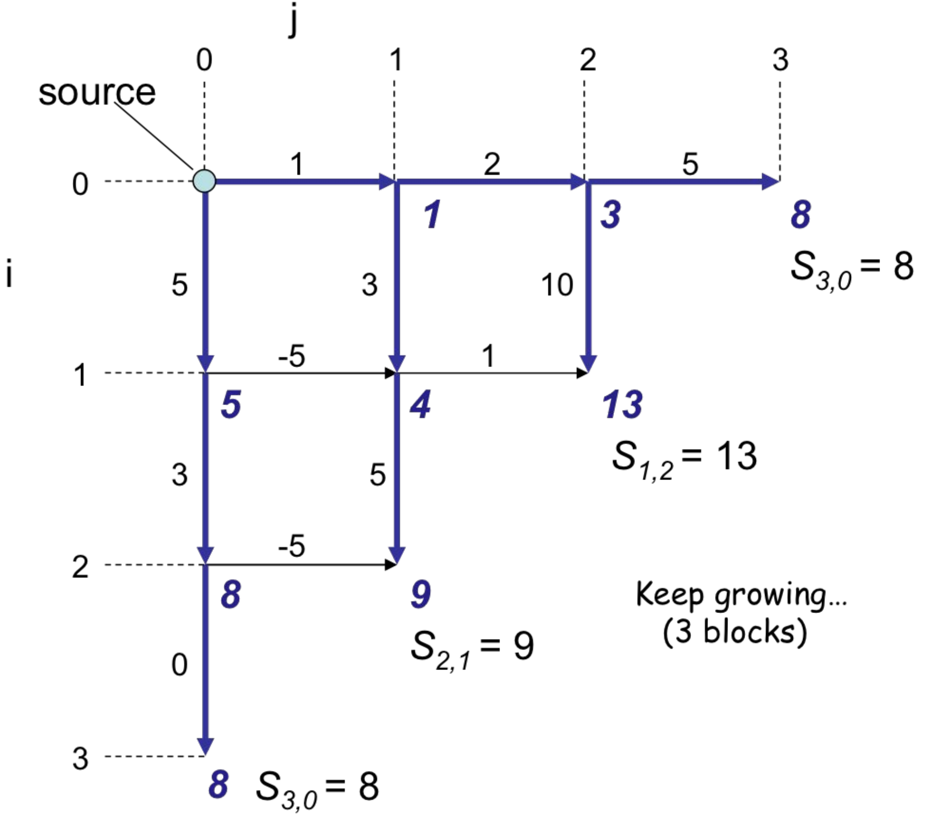
Then grow the solution a block at a time while tabulating the results for each intersection

$S_{1,1} = 4$

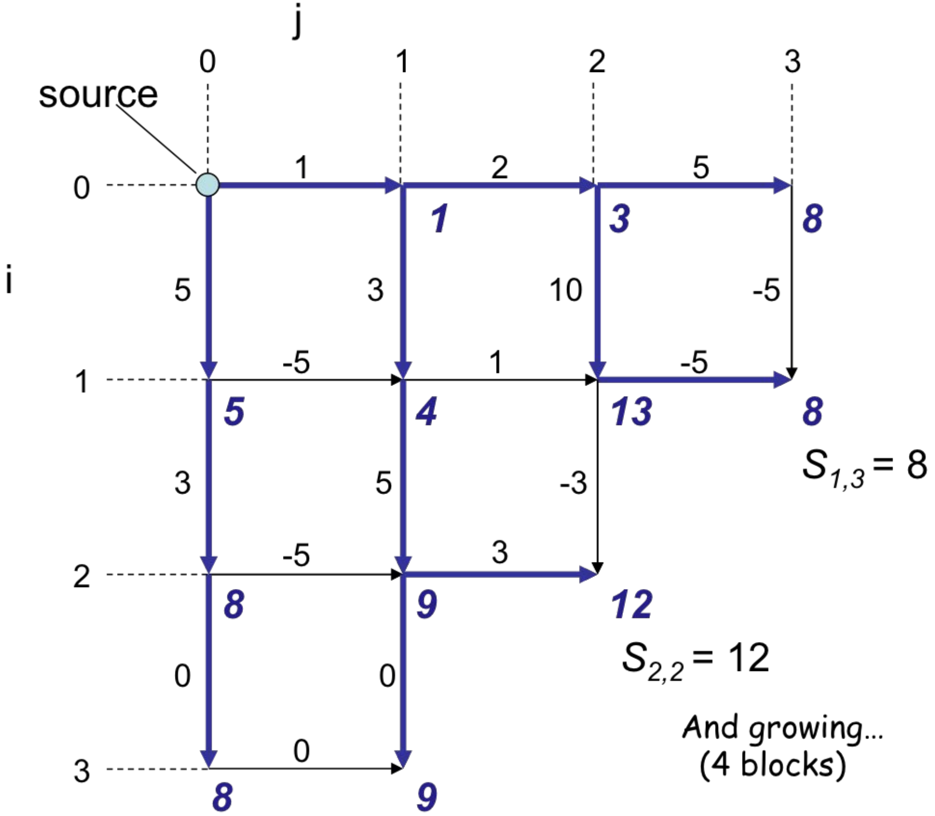
Note: We'll use our table to keep track of two things. The value of the best path to the given intersection, and the direction from where it came

MTP: Dynamic Program Continued

For each intersection let's keep track of the score and the direction that our "best" answer came from… We could do this by putting a yellow sticky on a corner lamp post, which said we saw *N* sites and arrived from either the north or the west



MTP: Dynamic Program Continued

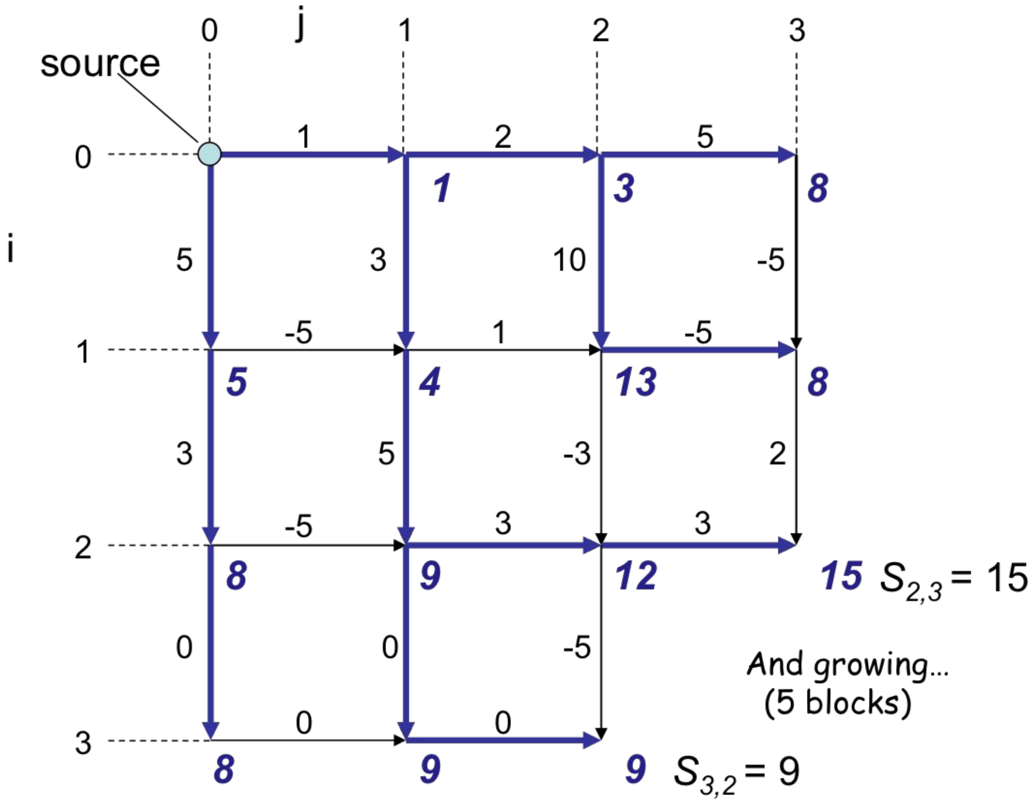


$S_{1,3} = 8$

$S_{2,2} = 12$

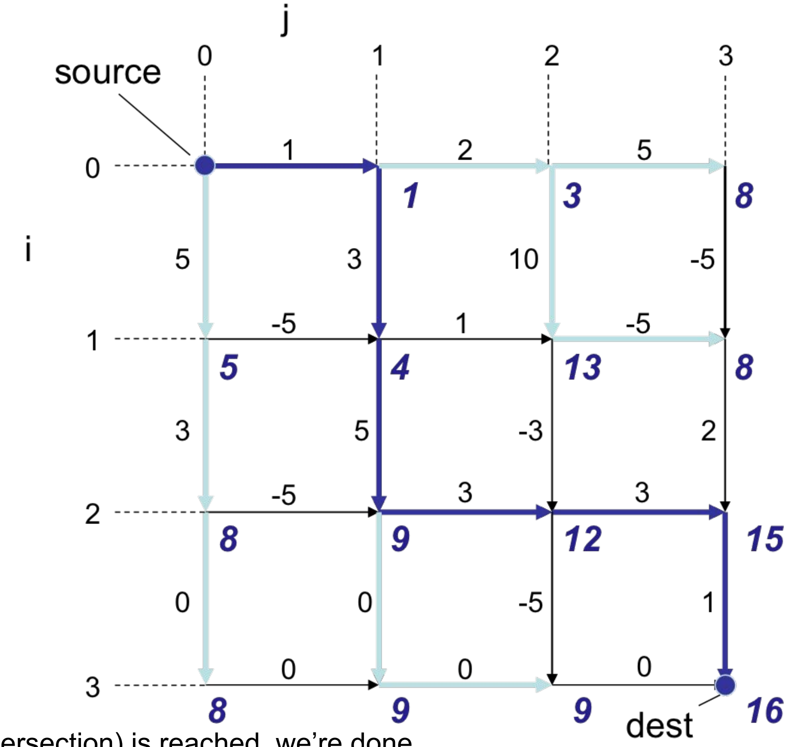
And growing...
(4 blocks)

MTP: Dynamic Program Continued



And growing…
(5 blocks)

MTP: Dynamic Program Continued



- Once the *destination* node (intersection) is reached, we’re done.
- Our table will have the answer of the maximum number of attractions stored in the entry associated with the destination.
- We use the *links* back in the table to recover the path. (Backtracking)

MTP: Recurrence

Computing the score for a point (i,j) by the recurrence relation:

Path to the intersection from the left

$$s_{i,j} = \max \begin{cases} s_{i-1,\,j} + \text{weight of the edge between } (i\text{-}1,\, j) \text{ and } (i,\, j) \\ s_{i,\,j-1} + \text{weight of the edge between } (i,\, j\text{-}1) \text{ and } (i,\, j) \end{cases}$$

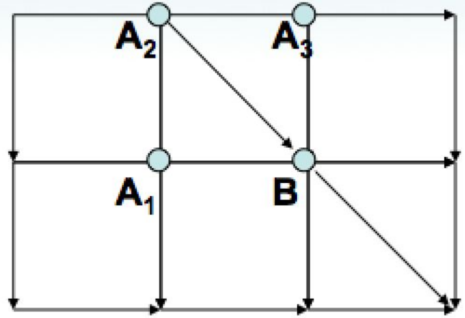
Path to the intersection from above

The running time is *nm* for a n × m grid

- You visit all intersections once, add two numbers, compare which is larger, save it and it's direction

(n = # of rows, m = # of columns)

Manhattan Is Not A Perfect Grid



What about diagonals?

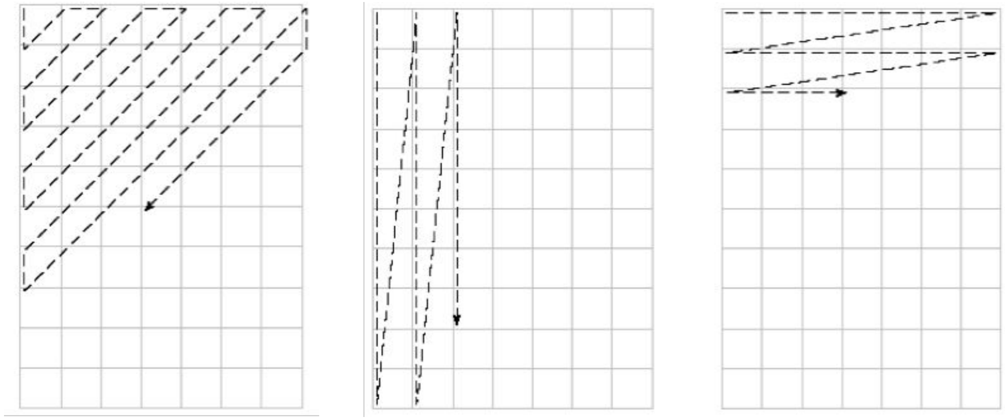
Broadway, Greenwich, etc.

- Easy to fix. Just adds more recursion cases.
- The score at point B is given by:

$$s_B = \max \begin{cases} s_{A1} + \text{weight of the edge } (A_1, B) \\ s_{A2} + \text{weight of the edge } (A_2, B) \\ s_{A3} + \text{weight of the edge } (A_3, B) \end{cases}$$

Other ways to safely explore the Manhattan

- We chose to evaluate our table in a particular order.
  Uniform distances from the source (all points one block away, then 2 blocks, etc.)
- Other strategies:
  - Column by column
  - Row by row
  - Radiate out along diagonals
- This choice can have performance implications



Next Time

- Return to sequence alignment
- Coding dynamic programs