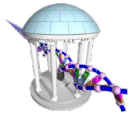


Comp 555 - BioAlgorithms - Spring 2018

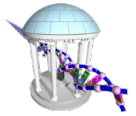


Q5E940_BOVIN	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_HUMAN	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_MOUSE	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_RAT	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_CHICK	-----MPREDRATWKSNYFMKIIQLDDYPKCFVVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_RANSY	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
Q7ZUG3_BRARE	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0 ICTPU	-----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0_DROME	-----MVRENKAWKAQYFKIVVLFDFEFPKCFIVGADNVGSKOMQOIRMSLRGK-AVVLGKNTMMRKAIRGHLENN--PALE	76
RLA0 DICDI	-----MSGAG-SKRKKLFIEKATKLFETTDKMIIVAEADNVGSSQLQKIRKSIIRGI-GAVLMGKNTMIRKVIIRDLDLADSK--PELD	75
Q54LP0_DICDI	-----MSGAG-SKRKNVFIEKATKLFETTDKMIIVAEADNVGSSQLQKIRKSIIRGI-GAVLMGKNTMIRKVIIRDLDLADSK--PELD	75
RLA0_PLAFB	-----MAKLSKQKQKQMYIEKLSLQIQVSKILVHVVDNVGSSQMASVRSLSRGK-ATILMGKNTIRRTALKKNLQAV--PQIE	76
RLA0_SULAC	-----MIGLAVTTTKKIAKWKVDEVAELTEKLLTKHTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLNFNIALKNAG----YDTK	79
RLA0_SULTO	-----MRIMAVITQERKIAKWKIEEVKLEEKLEXYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNLLFGIAAKNAG----LDVS	80
RLA0_SULSO	-----MKRLALALQKRVASWKLEEVKELTELTKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNLLFKIAAKNAG----LDIE	80
RLA0_AERPE	-----MSVVSIVGQMYKREKIDPEWKTMLMLELELFSKIRVVVLFADLTCTPFFVVRVVRKLLWKK-YPMVAKKRIILKAMKAAGLE--LDDN	86
RLA0_PYRAE	-----MMLAIGKRRYVTRQVYPAKRVKIVSEATELLQKQYVYVFLFDLHGLSSRIILHEVRYRLRRY-GVIKTIKPTLFAFTKVVYGG----IPAE	85
RLA0_METAC	-----MAEERHHTHTPQWKKDEIENIKELIQSHKVFQMGVIEGILLATKMKOIRRDLDKDV-AVLKVSRRNTLLERALNQLG----ETIP	78
RLA0_METMA	-----MAEERHHTHTPQWKKDEIENIKELIQSHKVFQMGVIEGILLATKMKOIRRDLDKDV-AVLKVSRRNTLLERALNQLG----ESIP	78
RLA0_ARCFU	-----PPEYKRVAVEEIKRMISSEVVAIVSFRNVFVAGQMKIRREFRQK-AEIKVTKNLLERALDALG----GDYL	75
RLA0_METKA	-----MAVKAKGQPPSGYEKVAEKKRREVEKLEKLMDEYENVGLVDLEGPAPLOEIRAKLREDDTIIRMSRRNTLMRALALEEKIDER--PELE	88
RLA0_METTH	-----MAHVAEWKKEVEQVLDLIRKYEVEVGIANLADIPAROLOKMRQTLRDS-ALIRMSKKTLLISLALEKAGREL--ENVD	74
RLA0_METTL	-----MITAESEHKIAPWKIEEVNKLKELLKNGQIVAVLDMMVEVPAROLOEIRDKIR-ETMLKMSRRNTLLERALKEVAEETGNPEFA	82
RLA0_METVA	-----MIDAKSEHKIAPWKIEEVNALKELLKNSANVIALIDMMVEVPAROLOEIRDKIR-DQMLKMSRRNTLLERALKEVAEETGNPEFA	82
RLA0_METJA	-----METKVAHVAPWKIEEVKTLKGLIKSKPVAIVDDMDVAPLOEIRDKIR-DKVKLRMSRRNTLLERALKEAAEELNPKLA	81
RLA0_PYRAB	-----MAHVAEWKKEVEEELANLKSYPVIALVDVSSMPAYPLSQMRRLLIRENGGLLRSRRNTLLELAIKKAAGELGKPELE	77
RLA0_PYRHO	-----MAHVAEWKKEVEEELAKLKSYPVIALVDVSSMPAYPLSQMRRLLIRENGGLLRSRRNTLLELAIKKAAGELGKPELE	77
RLA0_PYRFU	-----MAHVAEWKKEVEEELANLKSYPVIALVDVSSMPAYPLSQMRRLLIRENNGGLLRSRRNTLLELAIKKAAGELGKPELE	77
RLA0_PYRKO	-----MAHVAEWKKEVEEELANIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRSRRNTLLELAIKKAAGELGQPELE	76
RLA0_HALMA	-----MSAESEKRTETIPQWKQEEVDATVEMIESYEVGVDNACIPSRLODMRRDLHGT-AELRSRRNTLLELALDDVD--DGLE	79
RLA0_HALVO	-----MSESEVRQTEVTPQWKREEVDELVDVIESYEVGVDVAGIPSRLODMRRDLHGS-AAVRSRRNTLVNRADEVN--DGFE	79
RLA0_HALSA	-----MSAEEQRTTEVTPQWKQEEVDELVDLLETVDYSGVGVNVTGIPSRLODMRRDLHGQ-AALRSRRNTLVRALEEAG--DGLD	79
RLA0_THEAC	-----MKEVTSQKKELVNEITDRIKASRSVAIVDAGIRTRQIOTIRGKNRGK-INLKVKKLLFLKALENLGD--EKLS	72
RLA0_THEVO	-----MRKINPKKKEIVSELAQDITKSKAVAVIDKIGVTRROMODIRAKNRDK-VKIKVVKLLFLKALDSIND--EKLT	72
RLA0_PICTO	-----MTPEAQKIDFVKNLENEINRSKVAIVSIVKCLRNNEFKIRNSIRDK-ARIKVSRRALLRLAIENTGK--NNIV	72
ruler	1.....10.....20.....30.....40.....50.....60.....70.....80.....90	

- How well do our methods of mapping spectrums to sequences scale?
- How can we determine a peptide's sequence in the presence of errors or impurities?

Comparing Sequences

Sequence Similarity

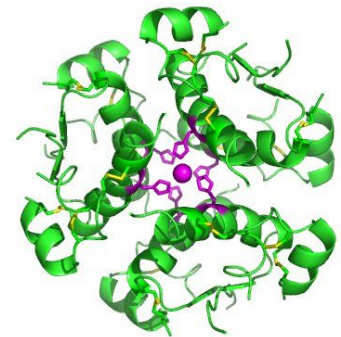


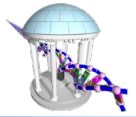
- A common problem in biology

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 : MALWtRLLPLLALLAlWAPAPAqAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAnpQagaVEL_Ggg1__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.

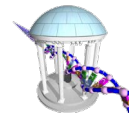
The Alignment Game



Let's consider only 2 sequences, and establish “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 = $\text{Max}(\text{len}(\text{top}), \text{len}(\text{bot}))$
 - Maximum moves = $\text{len}(\text{top}) + \text{len}(\text{bot})$
- How many possible moves?
 - Less than $3^{\text{len}(\text{top}) + \text{len}(\text{bot})}$
- How big for our problem instance?
 - $\text{len}(\text{Human}) = 98$, $\text{len}(\text{dog}) = 110$
 - $3^{208} \approx 1.73 \times 10^{90}$, almost a googol (not a google)
- What algorithm solves this problem?
 - Make each move by considering only a short horizon following the current alignment thus far

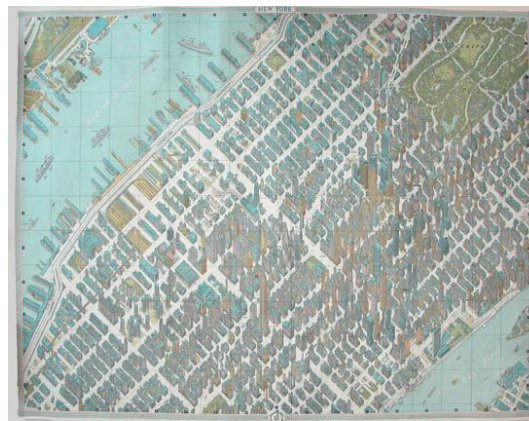


There is an efficient solution



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

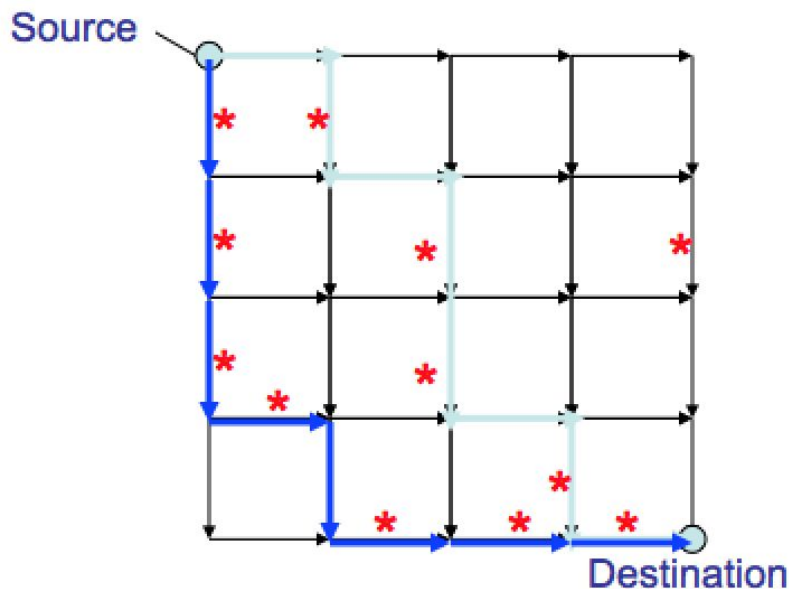
Finding a best city tour on a Manhattan grid



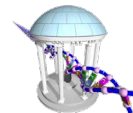
Manhattan Tourist Problem (MTP)



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

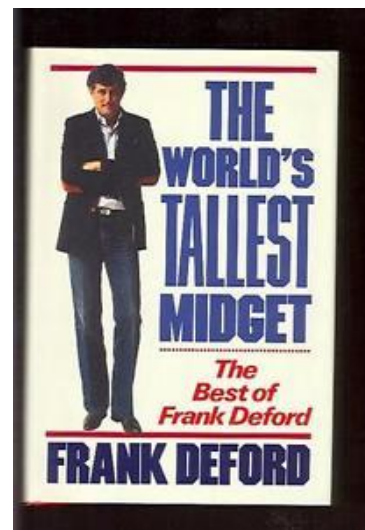


Goal: Find the maximum weighted shortest path in a grid.

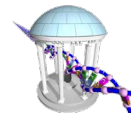
Input: A weighted grid G with two distinct vertices, one labeled *source* and the other labeled *destination*

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

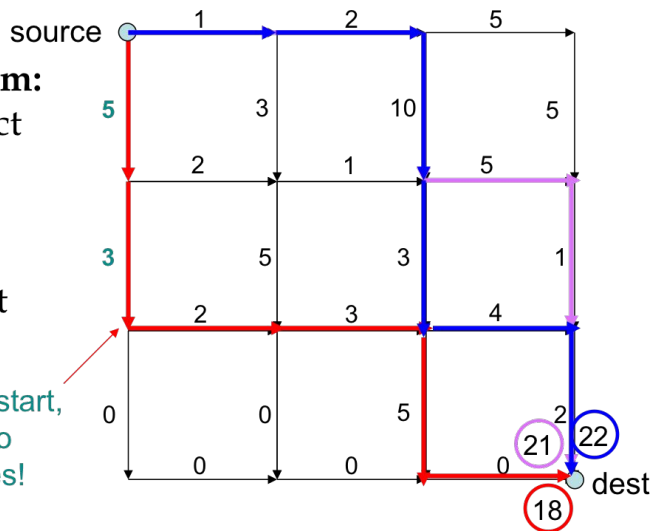


Greedy Algorithm:

At each step select the maximum weight block.

Greedy has a short horizon

promising start,
but leads to
bad choices!



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 go towards the street with the most attractions

A New Solution Strategy

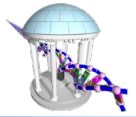


Dynamic Programming is a technique for *computing recurrence relations efficiently by storing and reusing intermediate results*

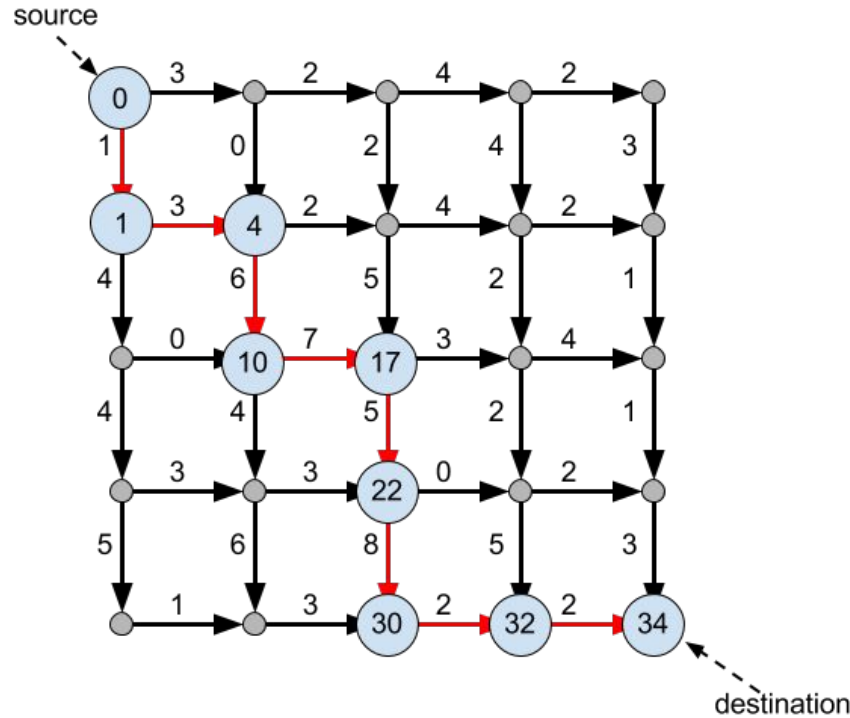
Three keys to constructing a dynamic programming solution:

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 the needed partial results

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

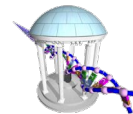


MTP Dynamic Program Solution



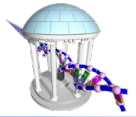
The solution may not be unique, but it will have the best possible score

MTP Dynamic Program Strategy



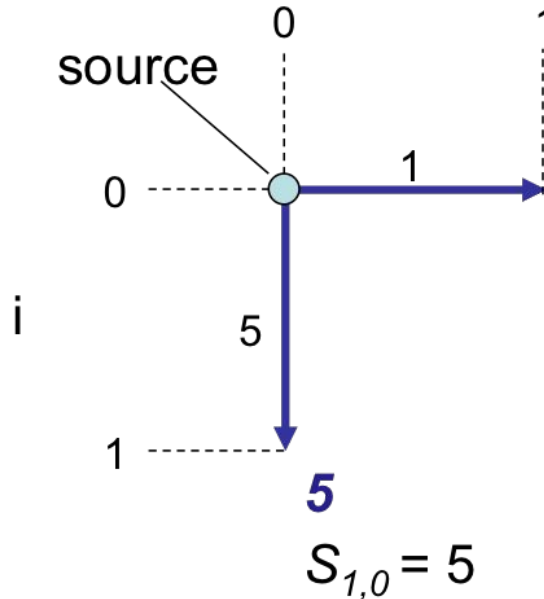
- Instead of solving the Manhattan Tourist problem directly, (i.e. the path from $(0,0)$ to (n,m)) we will solve a more general problem: find the longest path from $(0,0)$ to any arbitrary vertex (i,j) .
- If the longest path from $(0,0)$ to (n,m) passes through some vertex (i,j) , then the path from $(0,0)$ 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

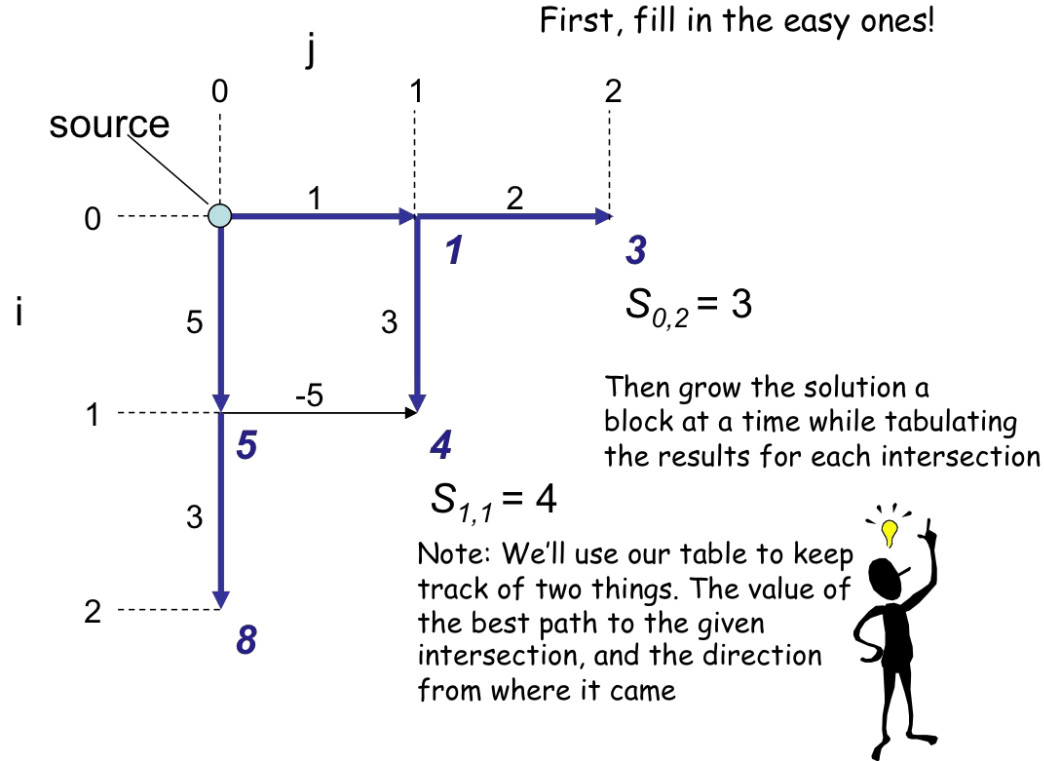


First, fill in the easy ones!
Those 1 block
from the source

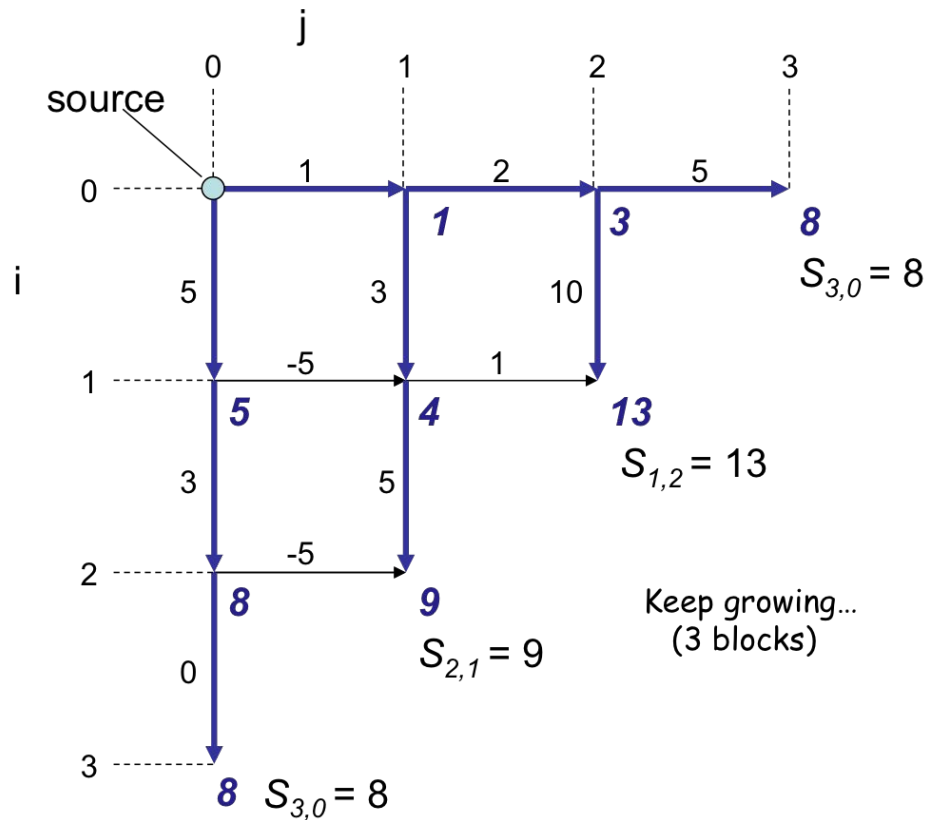
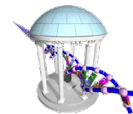




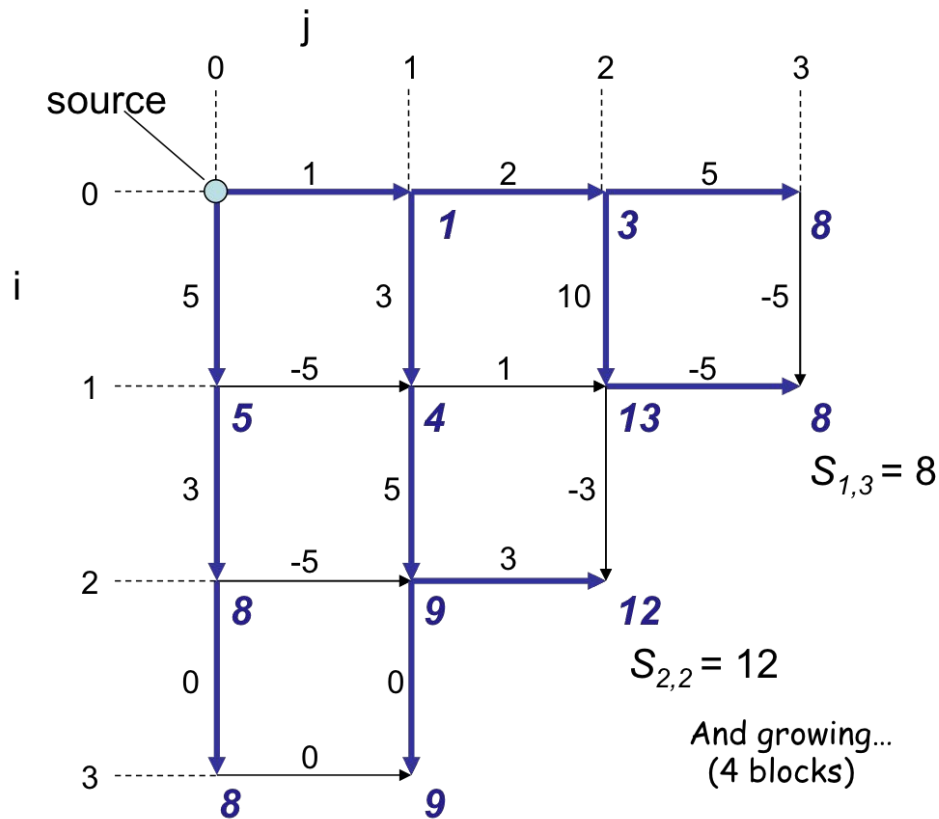
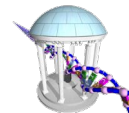
MTP: Dynamic Program Continued



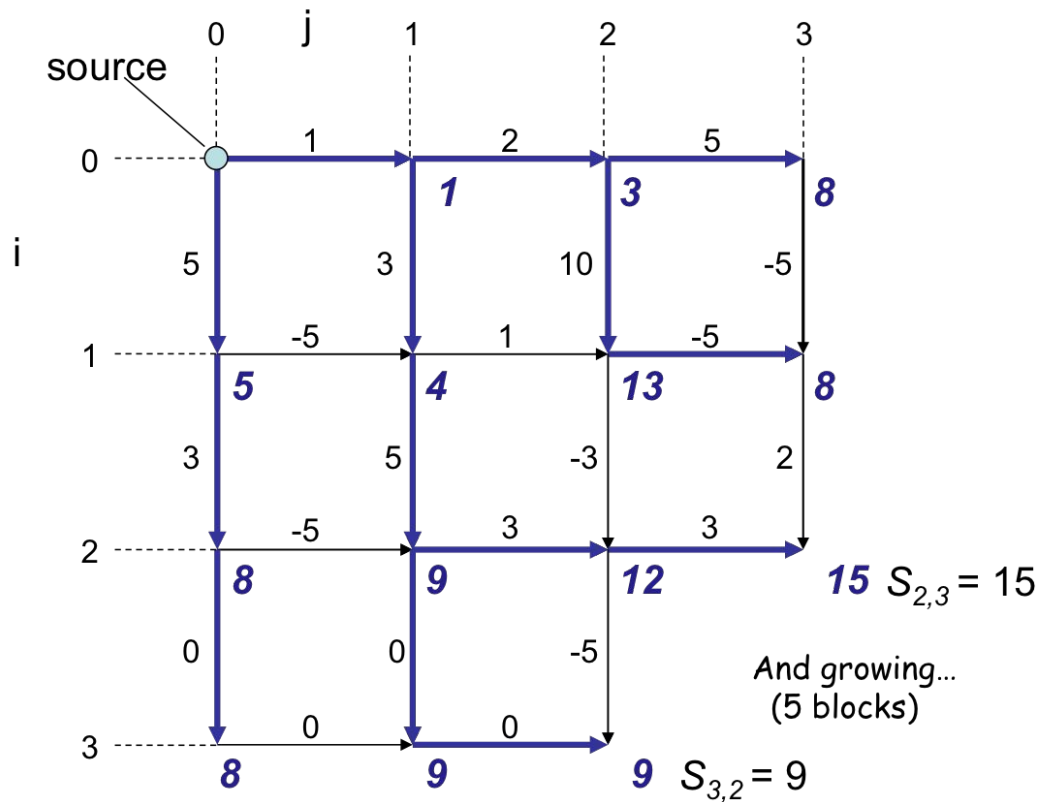
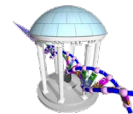
MTP: Dynamic Program Continued

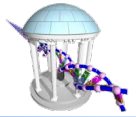


MTP: Dynamic Program Continued

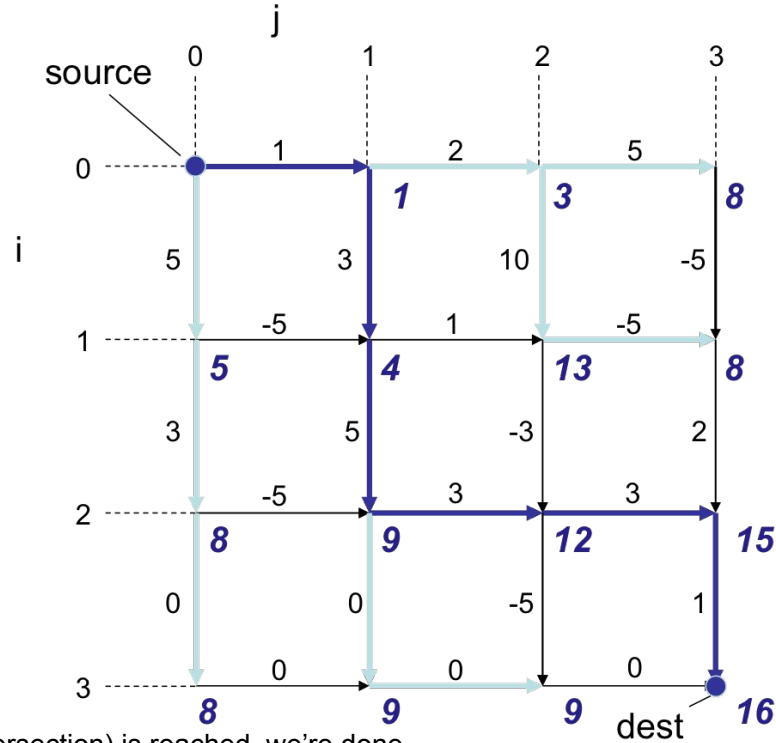


MTP: Dynamic Program Continued

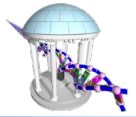




MTP: Dynamic Program Continued



- 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:

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

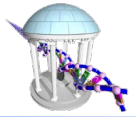
Path to the intersection from the left

Path to the intersection from above

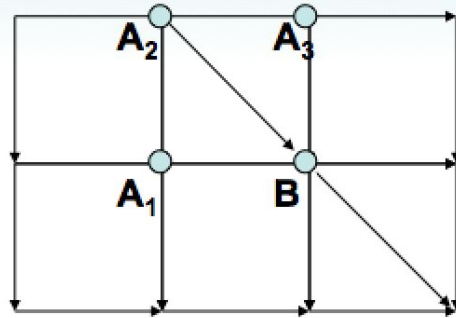
The running time is nm for a $n \times m$ grid

- (You visit all intersections once, and perform 2 tests)

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



Manhattan Is Not A Perfect Grid

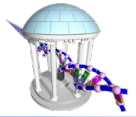


What about diagonals?

Broadway, Greenwich, etc.

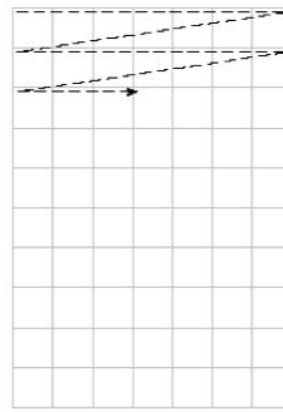
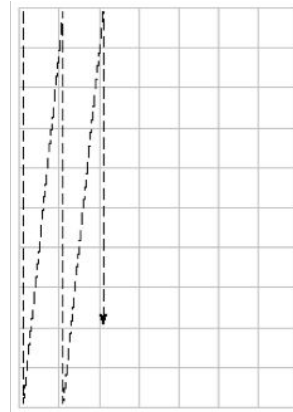
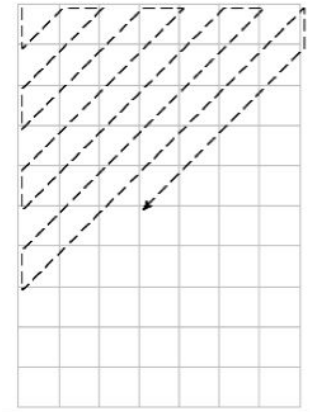
- Easy to fix. Just adds more recursion cases.
- The score at point B is given by:

$$s_B = \max \left\{ \begin{array}{l} s_{A_1} + \text{weight of the edge } (A_1, B) \\ s_{A_2} + \text{weight of the edge } (A_2, B) \\ s_{A_3} + \text{weight of the edge } (A_3, B) \end{array} \right.$$

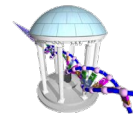


Other ways to safely explore the Manhattan

- 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

