

Comparing Sequences

```

Q5E940 BOVIN -----M*PREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0_HUMAN -----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0_MOUSE -----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0_RAT -----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0_CHICK -----MPREDRATWKSNYFMKIIQLDDYPKCFVVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0_RANSY -----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--SALE 76
Q7ZUG3 BRARE -----MPREDRATWKSNYFLKIIQLDDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0 ICTPU -----MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKMQQIRMSLRGK-AVVLGMGKNTMMRKAIRGHLENN--PALE 76
RLA0 DROME -----MVRENKAAWKAQYFIKVVVLFDEFPKCFIVGADNVGSKMQQIRMSLRGL-AVVLGMGKNTMMRKAIRGHLENN--PQLE 76
RLA0 DICDI -----MSGAG-SKRKKLFIEKATKLFITTDKMIVAEADVFVSSQLQKIRKSIRGI-GAVLMGKKIMIRKVIIRDLADSK--PELD 75
Q54LP0 DICDI -----MSGAG-SKRKNVFEKATKLFITTDKMIVAEADVFVSSQLQKIRKSIRGI-GAVLMGKKIMIRKVIIRDLADSK--PELD 75
RLA0 PLAF8 -----MAKLSKQKKQMYTEKLSLILQYSKLILVHVVDNVGSNMQMASYRKSIRGK-ATLLMGKNTIRRTALKKNLQAV--PQLE 76
RLA0 SULAC -----MIGLAVTTTKKIAKWKYDEVAELTEKIKTKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNLFNIAAKNAG----YDLEK 79
RLA0 SULTO -----MRIMAVITQERKIAKWKIEEVKELEKLRVHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNLFNIAAKNAG----LDVS 80
RLA0 SULSO -----MKRLALALKQKVASWKEEVKELTELKNSNTLLIGNLEGGPADKLHEIRKKLRGK-ATIKVTKNLFNIAAKNAG----IDLE 80
RLA0 AERPE MSVVSIVGQMYKREKIPDEWKTLMLELEELFSKRVVLFADLTGPTFVVQRVKRLWKK-YPMVVAKKRIILRAMKAAGLE--LDDN 86
RLA0 PYRAE -MMLATIGKRRYVRTQYYPARKVKIVSEATELLQKYPYVFLDHLGSLRILHEHYRRLRY-GVIKTIKPTLFKIAFTKVYGG--IPAE 85
RLA0 METAC -----MAEERHTEHIPQWKDEIENIKELIQSHKVFQMVIEGILATKMKIRRDLDKV-AVLKVSRLNLELRAALNQLG----ETIP 78
RLA0 METMA -----MAEERHTEHIPQWKDEIENIKELIQSHKVFQMVIEGILATKMKIRRDLDKV-AVLKVSRLNLELRAALNQLG----ESIP 78
RLA0 ARCFU -----MAAVRGS--PDEYKVRAVEEIKRMISSKPVVAIVFRNVFAGQMQKIRREFRGK-AEIKVVKNTLELRAALDGLG--GDYL 75
RLA0 METKA MAVKAKGQPPSQYE PKVAEWKRRVEKLEKLMDEVENVGLVDEGIPAPQLOEIRAQLERDITIRMSRNTLMRIALEEKLEDER--PELE 88
RLA0 METH -----MAHVAEWKKEVEEQLHDLKGEVYVGIANLADIPARLOKMRQTLRDS-ALIRMSKNTLISLALKAAGREL--ENVD 74
RLA0 METTL -----MITRESEHKIADPKWIEEVNKLKELLNQOIALVDMMEVPARQLOEIRDKIR-CTMTLKMSRNTLIEIRAKEVAETGNPEFA 82
RLA0 METVA -----MIDAKSEHKIADPKWIEEVNALKELKSNVIALIDMMEVPAVQLOEIRDKIR-DQMTLKMSRNTLIEIRAKEVAETGNPEFA 82
RLA0 METJA -----METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAQLOEIRDKIR-DKVKLRMSRNTLIEIRAKEVAEELNPKLA 81
RLA0 PYRAB -----MAHVAEWKKEVEEELANLIKSPVVALVDVSSMPAYPLSQMRRLIRENGCLLRVSRNTLIELAIKKAAQELGKPELE 77
RLA0 PYRHO -----MAHVAEWKKEVEEELAKLIKSPVVALVDVSSMPAYPLSQMRRLIRENGCLLRVSRNTLIELAIKKAAQELGKPELE 77
RLA0 PYRFU -----MAHVAEWKKEVEEELANLIKSPVVALVDVSSMPAYPLSQMRRLIRENGCLLRVSRNTLIELAIKKAAQELGKPELE 77
RLA0 PYRKO -----MAHVAEWKKEVEEELANLIKSPVVALVDVAGVPAYPLSKMRDKLR-CKALLRVSRNTLIELAIKKAAQELGKPELE 76
RLA0 HALMA -----MSAESERKTETIPWKQEEVDIVEMIESYESVGVNVIAGIPSRQLDMRRDLHGT-AELRVSRNTLIEALDDVD----DGLE 79
RLA0 HALVO -----MSESEVQTEVLPQWKREVDDELVDIESVSVGVVYVAGIPSRQLDMRRDLHGT-AELRVSRNTLIEALDDVD----DGLE 79
RLA0 HALSA -----MSAEQRTTEVPEWKRQEVAVLVDLLETDSVGVVYVYVAGIPSRQLDMRRDLHGT-AALRMSRNTLLVRALEEAG--DGLD 79
RLA0 THEAC -----MKEVSQQKELVNEITTRIKASRSVAIVDLAGIRTRIODIRGKNRGK-INLKVYKILLLFKALENLCD----EKLS 72
RLA0 THEVO -----MRKINPKKKEIVSELAQDITIKSVAIVDIKGVTRIQMODIRAKNRDK-VKIKVYKILLLFKALDSIND----EKLT 72
RLA0 PICTO -----MTEPAQWKIDFVKNLENEINSRKAIVSILGLRNNFQKIRNSIRDK-ARIKVSRARLLRLALENLGK----NNIV 72
ruler 1.....10.....20.....30.....40.....50.....60.....70.....80.....90

```

By Miguel Andrade at English Wikipedia

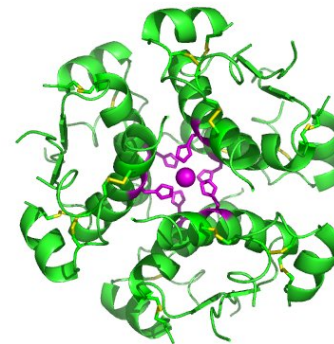
Sequence Similarity

- A common problem in Biology

Insulin Protein Sequence

Human	MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQGSLQPLALEGSLQKRGIVEQCCTSI	CSLYQLENYCN
Dog	MALWMRLLPLLALLALWAPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEDLQVRDVELAGAPGEGGLQPLALEGALQKRGIVEQCCTSI	CSLYQLENYCN
Cat	MAPWTRLLPLLALLSLWIPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEDLQGKDAELGEAPGAGGLQPSALEAPLQKRGIVEQCCASVCSLYQLEHYCN	
Pig	MALWTRLLPLLALLALWAPAPAQAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAENPQAGAVELGGGLGGLQALALEGPPQKRGIVEQCCTSI	CSLYQLENYCN

- 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_____GSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
      |||
Dog : MALWMRLLPLLALLALWAPAPtRAfVNQHLCGSHLVEALYLVCGERGFFYTPKARREvEDLQvrDVELaG_APGeGGLQPLALEGA_LQKRGIVEQCCTSICSLYQLENYCN
      |||
Cat : MApWtRLLPLLALLsLWiPAPtRAfVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEDLQgkDaEL_GeAPGaGGLQPsALE_APLQKRGIVEQCCaSvCSLYQLEHYCN
      |||
Pig : MALWtRLLPLLALLA|WAPAPAqAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEmpQagaVEL_Ggg|_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 simplify the problem a bit by considering only 2 sequences, and establishing 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

Human ▾ Dog ▾ Start Score: 0

MALWMRLLPLLALLALWGPDPAAA

MALWMRLLPLLALLALWAPAPTRA

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 $O(3^{\text{len}(\text{top}) + \text{len}(\text{bot})})$
- How big are these for our problem instance?
 - $\text{len}(\text{Human}) = 98, \text{len}(\text{dog}) = 110$
 - $3^{208} \approx 1.73 \times 10^{99}$, 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



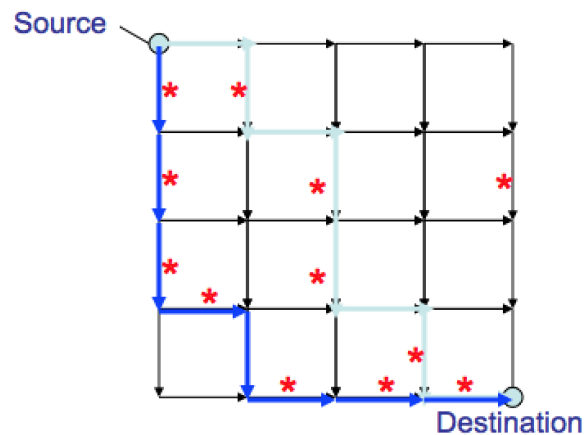
There is an efficient solution

- It relies on a rather surprising idea
 - The best score can be found for the $len(top)$ 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.



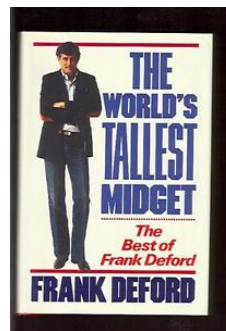
Manhattan Tourist Game

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

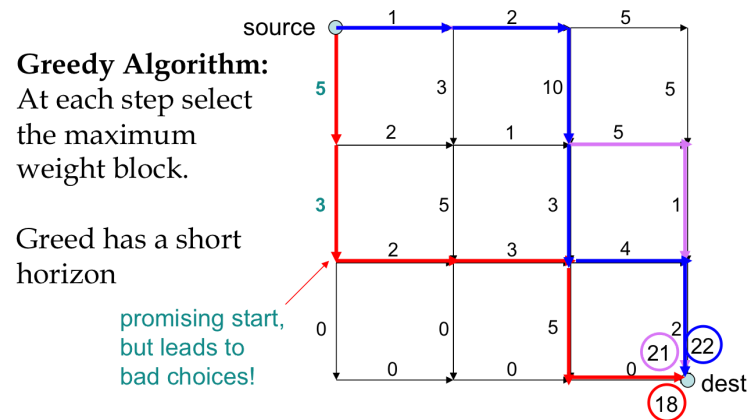
Input: A weighted grid G with two distinct vertices, one labeled *source* and the other labeled *destination*

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



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

MTP: Observations

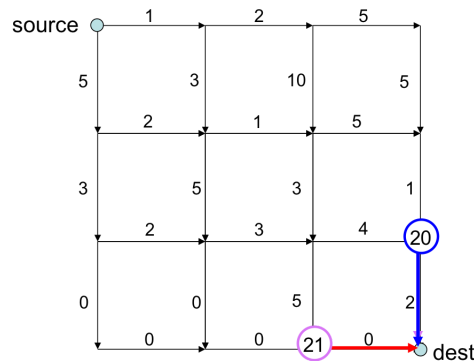
- There are limited number of ways to reach any destination
 - For example, in our grid, one can reach the destination node, (n,m) , from either the north, $(n,m-1)$, or the west $(n-1,m)$.
 - for each of those routes there is a known number of sites to see, so the best path is:

$$Score(n, m) = \text{Max}(Score(n - 1, m) + Edge(n - 1, m), Score(n, m - 1) + Edge(n, m - 1))$$

- Why is there only one edge per intersection? Because only one direction makes progress to our goal
- This rule applies recursively with the base case

$$Score(0, 0) = 0$$

- We could write this strategy as a recursive algorithm, but it would still not be efficient. Why?



A New Solution Strategy

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

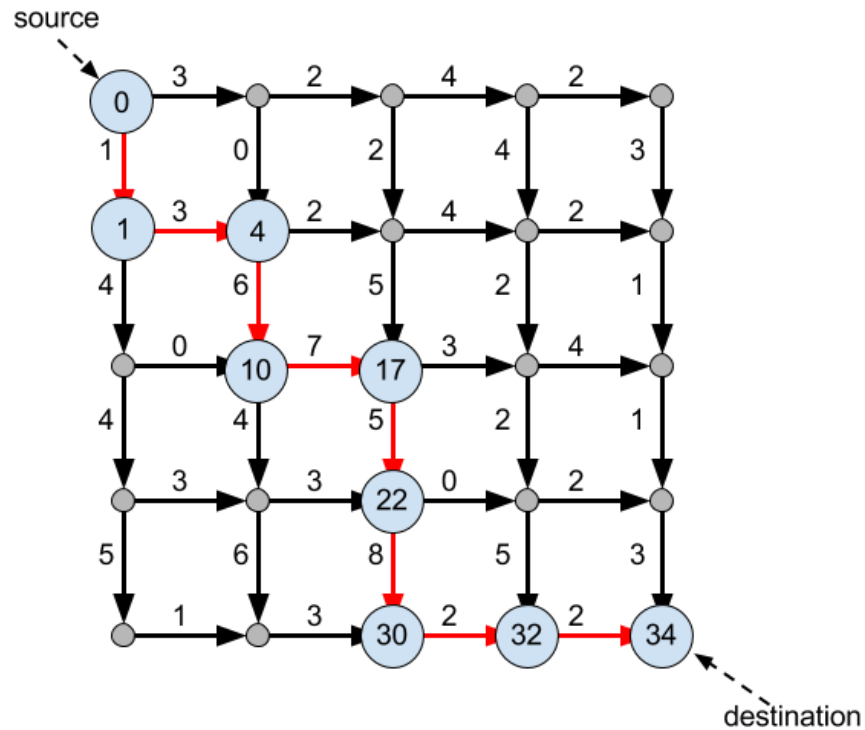
Three keys to constructing a dynamic programming solution:

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).
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

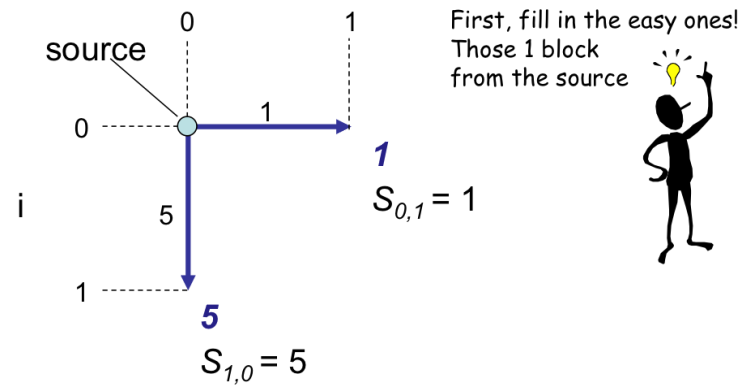
MTP Dynamic Program Strategy

- 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.

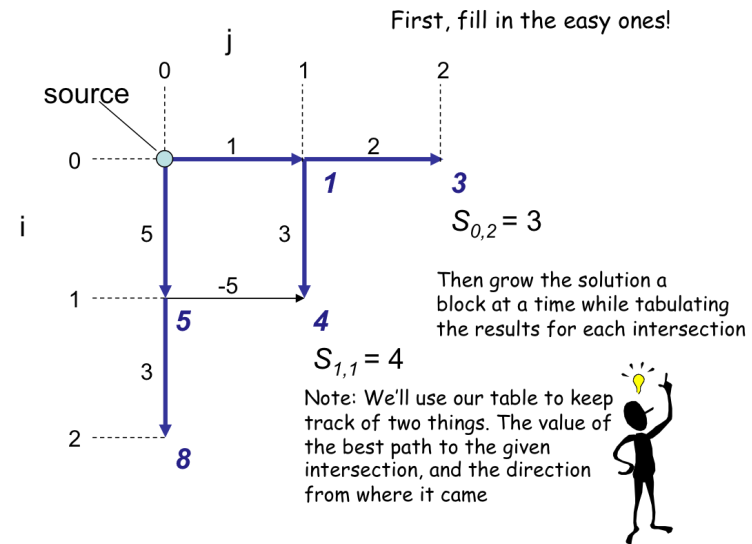


MTP: Dynamic Program

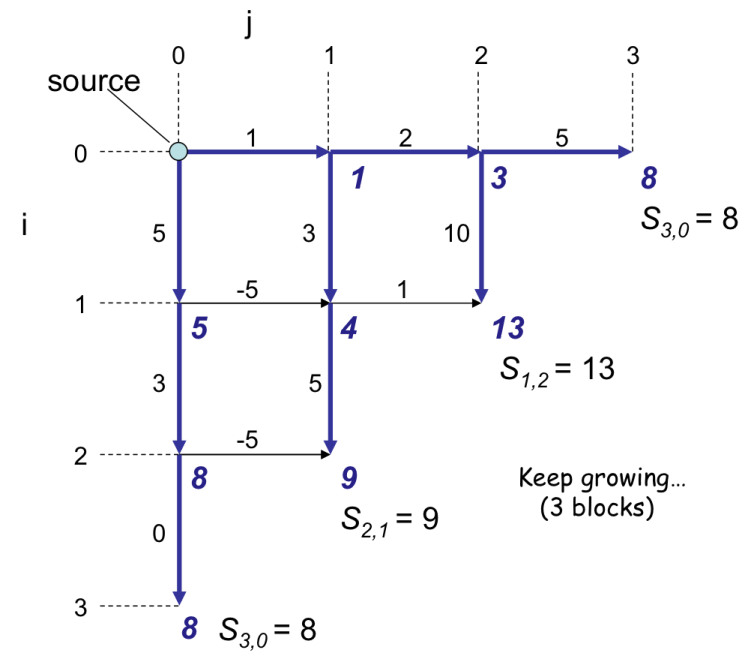
- Calculate optimal path score for *every* vertex in the graph between our source and destination
- Each vertex's score is the maximum of the prior vertices score plus the weight of the connecting edge in between



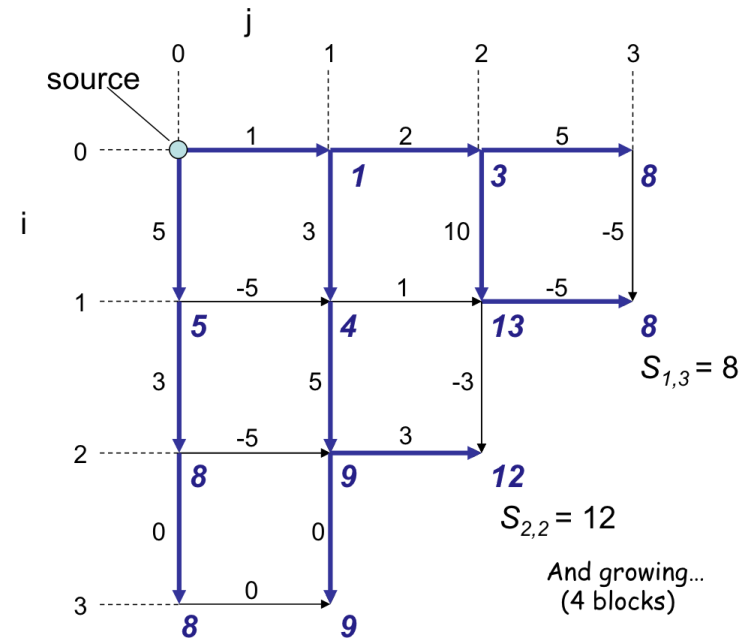
MTP: Dynamic Program Continued



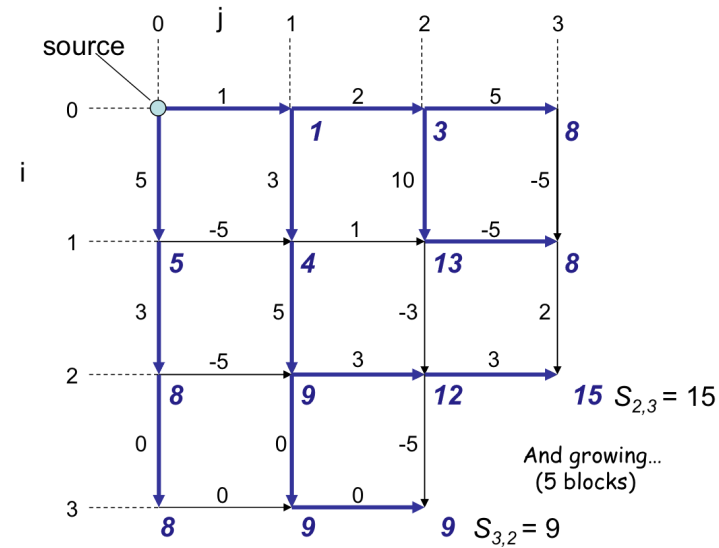
MTP: Dynamic Program Continued



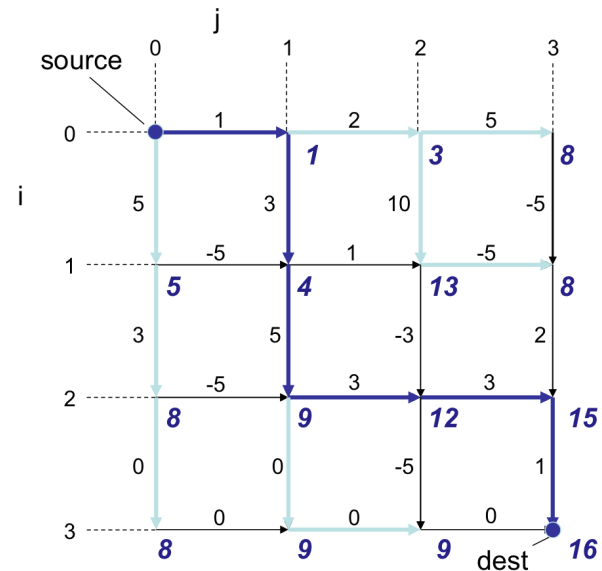
MTP: Dynamic Program Continued



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

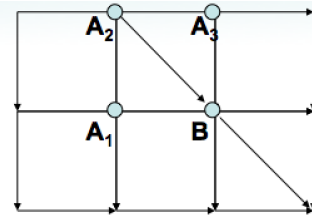
Path to the intersection from above

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?

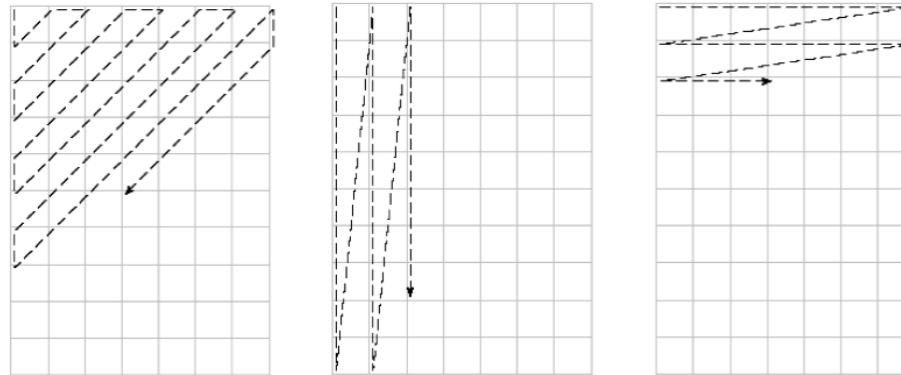
Broadway, Greenwich, etc.

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

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

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
 - Along diagonals
- This choice can have performance implications



Next Time (after the break)

- Return to biology
- Solving sequence alignments using Dynamic Programming

