Genome 540 discussion

February 4th, 2025
Joe Min

Genome Sciences
UNIVERSITY OF WASHINGTON

Agenda

Homework 4

Object oriented programming

Homework 4

Overview

1. Write a program to find the highest-weight path in a directed acyclic graph using dynamic programming
2. Run your program on a linked list created from DNA sequence

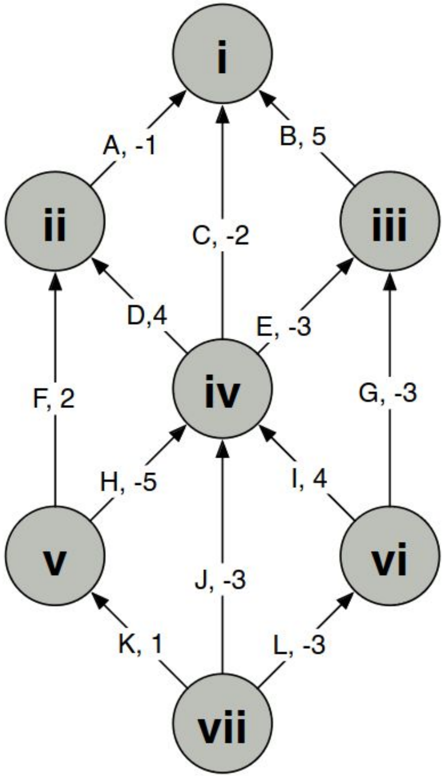
Homework 4: part 1

Read in and process a visual graph

First convert the visual graph into a text input file with vertices and edges

Then, find max weight path with dynamic programming

- With and without start/end constraints



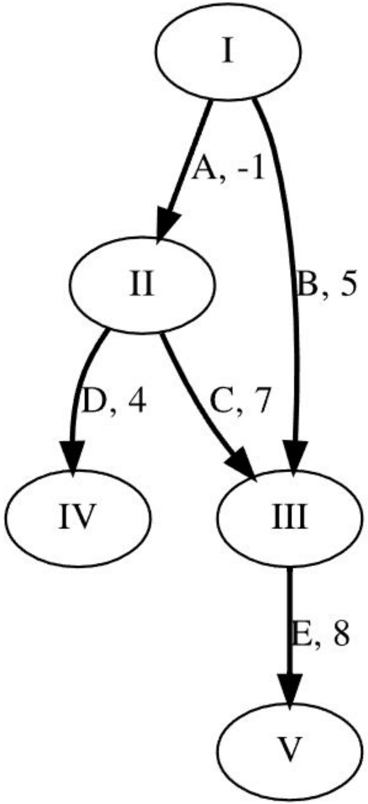

```
# INPUT FILE
V vii START
V vi
.
V i END
...
E A ii i -1
E B iii i 5
.
E L vii vi -3
```

```
# No constraints
Score: 8
Begin: vi
End: ii
Path: ID
```

```
# With constraints
Score: 4
Begin: vii
End: i
Path: LIDA
```

Dynamic programming example

```
# INPUT FILE
V I
V II
V III
V IV
V V
E A I II -1
E B I III 5
E C II III 7
E D II IV 4
E E III V 8
```



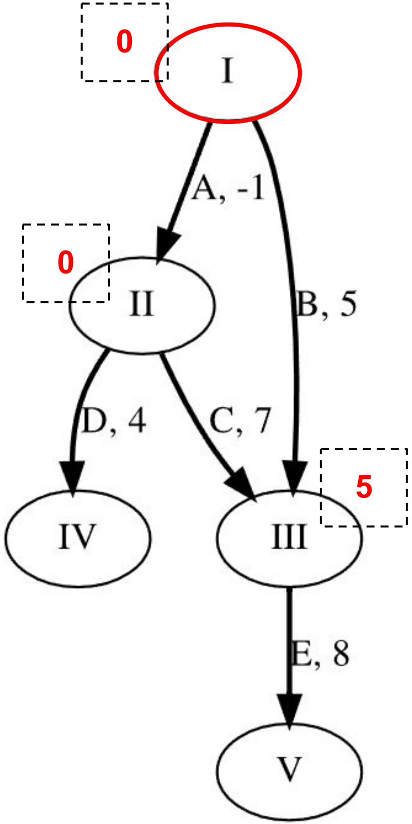
You can structure your input file in depth order so it's easier to read it in for bookkeeping

Vertex	I	II	III	IV	V
Highest weight parent	I	II	III	IV	V
Highest weight path weight	0	0	0	0	0

	End	Weight
Best overall path		

Dynamic programming example

```
# INPUT FILE
V I
V II
V III
V IV
V V
E A I II -1
E B I III 5
E C II III 7
E D II IV 4
E E III V 8
```



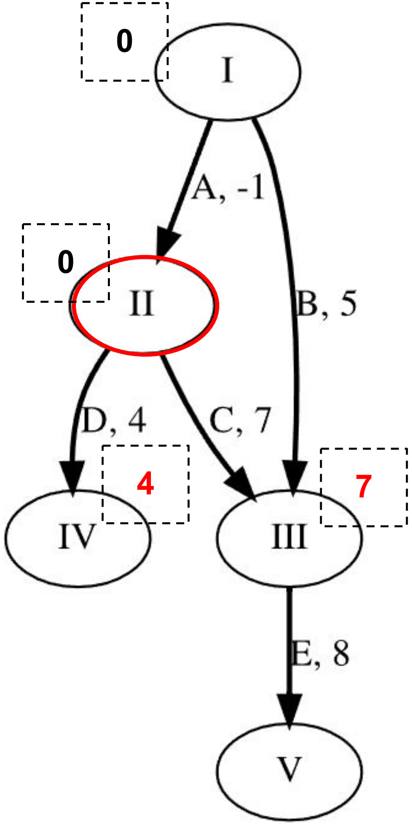
Process nodes (and their outbound edges) in depth order and update notes (as needed)

Vertex	I	II	III	IV	V
Highest weight parent	I	**II**	**I**	IV	V
Highest weight path weight	0	**0**	**5**	0	0

	End	Weight
Best overall path	**III**	**5**

Dynamic programming example

```
# INPUT FILE
V I
V II
V III
V IV
V V
E A I II -1
E B I III 5
E C II III 7
E D II IV 4
E E III V 8
```



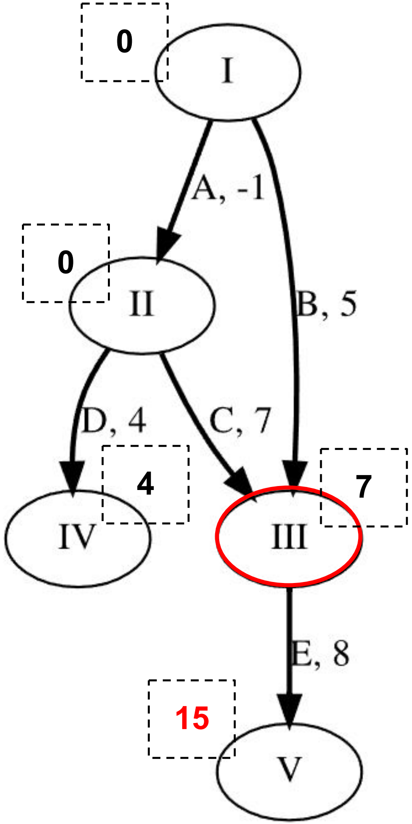
Continue down the graph in depth order

Vertex	I	II	III	IV	V
Highest weight parent	I	II	**II**	**II**	V
Highest weight path weight	0	0	**7**	**4**	0

	End	Weight
Best overall path	III	**7**

Dynamic programming example

```
# INPUT FILE
V I
V II
V III
V IV
V V
E A I II -1
E B I III 5
E C II III 7
E D II IV 4
E E III V 8
```



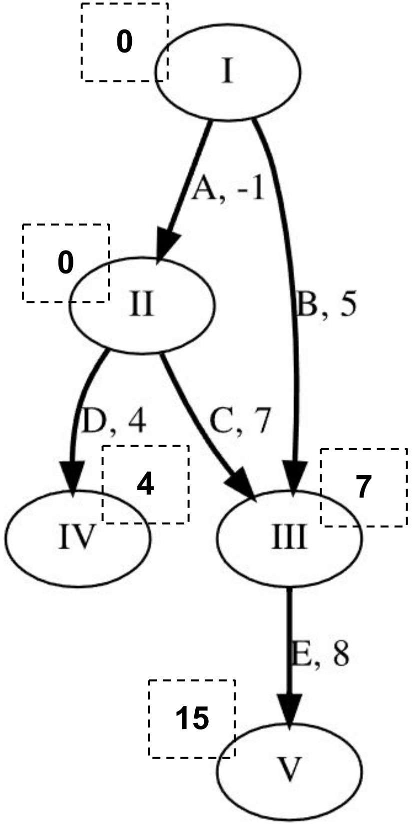
Once we run out of edges, we're done

Vertex	I	II	III	IV	V
Highest weight parent	I	II	II	II	**III**
Highest weight path weight	0	0	7	4	**15**

	End	Weight
Best overall path	**V**	**15**

Dynamic programming example

```
# INPUT FILE
V I
V II
V III
V IV
V V
E A I II -1
E B I III 5
E C II III 7
E D II IV 4
E E III V 8
```



We can work backwards to reconstruct the reversed path:

V → III → II

Vertex	I	II	III	IV	V
Highest weight parent	I	II	II	II	III
Highest weight path weight	0	0	7	4	15

	End	Weight
Best overall path	V	15

Homework 4: part 2

Overview

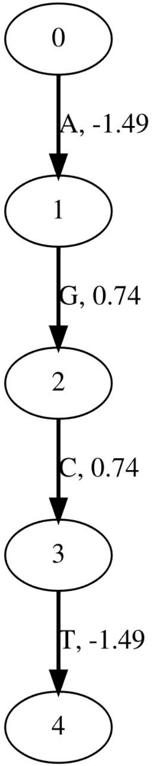
Create a linked list from a DNA sequence and a scoring scheme

- Positions are vertices
- Bases are edges

Run your program from part 1 on the graph

```
# GRAPH FILE
V 0
V 1
V 2
V 3
V 4
E A 0 1 -1.49
E G 1 2 .74
E C 2 3 .74
E T 3 4 -1.49
```

```
# SCORING FILE
A = -1.49
T = -1.49
G = .74
C = .74
```



Object oriented programming

What are objects?

Objects are instantiations of classes, which are data structures with custom functions

C++:

```cpp
class MyClass {        // The class
  public:              // Access specifier
    void myMethod() {  // Method/function defined inside the class
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

Python:

```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

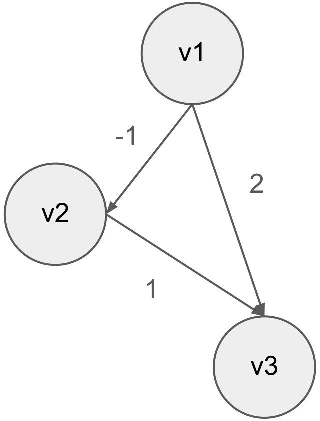
https://www.w3schools.com/cpp/cpp_class_methods.asp

What is object oriented programming?

Write programs in such a way that we primarily deal with objects that interact with each other within code, as opposed to more abstract representations of data

Example: graphs

To create a trivial graph and print out the neighbors of a single vertex, we can store value in a dictionary



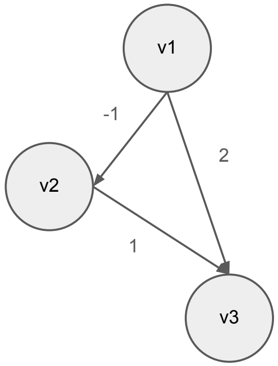

```
# Non-object oriented pseudocode
vertices = {}
for v_name in [‘v1’, ‘v2’, ‘v3’]:
        vertices[v_name] = []
for e_tuple in [(‘v1’, ‘v2’, -1), (‘v1, ‘v3’, 2), (‘v2’, ‘v3’, 1)]:
        v_name = e_tuple[0]
        v_neighbor = e_tuple[1]
        e_weight = e_tuple[2]
        vertices[v_name].append((v_neighbor, e_weight))
print([e[0] for e in vertices[‘v1’])
```

Example: graphs

```
# Pseudocode
class Vertex:
	# name is a string
	# edges is a list of Edge objects
	def __init__(self, name):
		self.name = name
		self.edges = []
	def add_edge(self, edge):
		self.edges.append(edge)
	def get_neighbors(self):
		neighbors = []
		for edge in self.edges:
			self.neighbors.append(edge.end)
		return neighbors

class Edge:
	# start and end are strings
	def __init__(self, start, end, weight):
		self.start = start
		self.end = end
		self.weight = weight
```

```
# Object oriented pseudocode
vertices = {}
for v_name in [‘v1’, ‘v2’, ‘v3’]:
	vertices[v_name] = Vertex(v_name)
for e_tuple in [(‘v1’, ‘v2’, -1), (‘v1, ‘v3’, 2), (‘v2’, ‘v3’, 1)]:
	v_name = e_tuple[0]
	vertex = vertices[v_name]
	edge = Edge(v_name, e_tuple[1], e_tuple[2])
	vertex.add_edge(edge)
print(vertices[‘v1’].get_neighbors())
```



Office hours

Reminder:

Homework 4 is due Sunday, February 9th at 11:59pm!