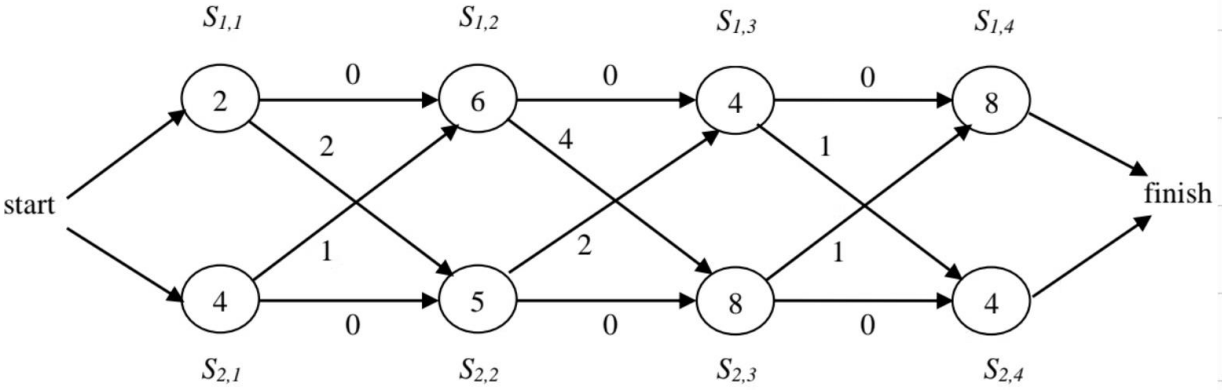
Question 1 (18 marks)

Suppose there are two assembly lines each with 4 stations, $S_{i,j}$. The assembly time is given in the circle representing the station and the transfer time is given next to the edge from one station to another.



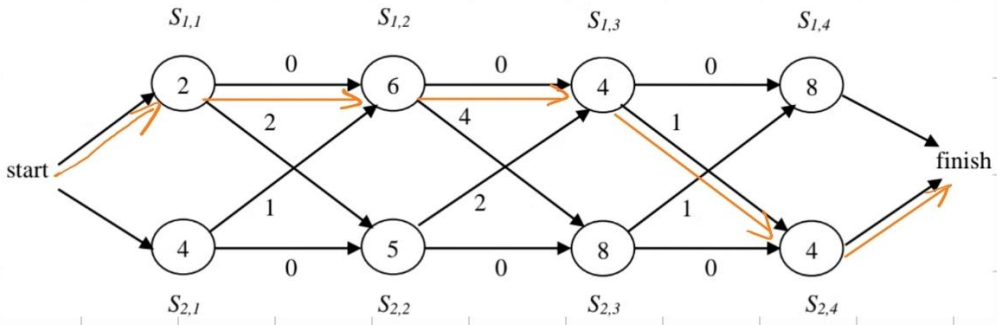
1. Using dynamic programming, fill in the table of the minimum time $f_i[j]$ needed to get through station $S_{i,j}$ and the *line* of the station just before $S_{i,j}$ on the fastest way to get through $S_{i,j}$. Show all the intermediate steps in computing these values. **(8 marks)**

1.

	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	End
From $S_{1,j}$ →		8	12	20	
From $S_{2,j}$ →		~~11~~	~~15~~	~~26~~	~~20~~
S_{1j}	2				
From $S_{1,j}$ →		9	~~20~~	17	
From $S_{2,j}$ →		~~9~~	17	~~21~~	17
S_{2j}	4				
	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	End

⇒

j	$f_{1,[j]}$	$f_{2,[j]}$
1	2	4
2	8	9
3	12	17
4	20	17



2. What is the minimum time f^* needed to get through the assembly line? **(2 marks)**

3. Based on the line information on the table, show how to find the fastest way (which stations should be chosen?) **(8 marks)**

2. Minimum time $f^* = 17$.

3. For each time j, choose the minimum station between $f_{1,[j]}$ and $f_{2,[j]}$. Then the fastest way will be found.

j	$f_{1,[j]}$	$f_{2,[j]}$
1	2	4
2	8	9
3	12	17
4	20	17

$\Rightarrow S_{1,1} \to S_{1,2} \to S_{1,3} \to S_{2,4}$ which is the fastest way.

Question 2 (14 marks)

1. Given a pattern CGTGC, create a shift table for letters A, G, C, T. **(4 marks)**
2. Apply Horspool's algorithm to search the pattern in text AGCCGTGC, what is the number of comparisons. **(10 marks)**

1. Pattern : C G T G C ⇒ shift table:

4 2 1

missing : A = 5

A	G	C	T
5	1	4	2

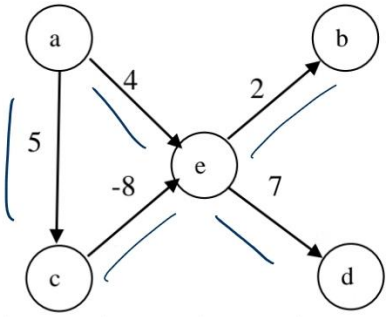
2. Horspool's algorithm:

									No.
T:	A	G	C	C	G	T	G	C	
P:	C	G	T	G	C	→ 1			+1
		C	G	T	G	C	→ 2		+1
			C	G	T	G	C		+5
									7

∴ The number of comparisons = 7.

Question 3 (16 marks)

For the following graph, run Bellman-ford algorithm to find all shortest paths from vertex *a*.



Initialization:

a	b	c	d	e
0	∞	∞	∞	∞

Iteration 1:

	a	b	c	d	e
a-e	0^{a}	∞	5^{a}	∞	∞
a-c	0^{a}	∞	5^{a}	∞	4^{a}
c-e	0^{a}	∞	5^{a}	∞	-3^{c}
e-b	0^{a}	-1^{e}	5^{a}	∞	-3^{c}
e-d	0^{a}	-1^{e}	5^{a}	4^{e}	-3^{c}
	a→c→e→b	a→c	a→c→e→d	a→c→e	

Iteration 2: No more updating

∴ The shortest path:

from a to b = -1, a→c→e→b

from a to c = 5, a→c

from a to d = 4, a→c→e→d

from a to e = -3, a→c→e

1. to find the optimal global alignment of AATG and AGC **(15 marks)**
2. to find the optimal local alignment of AATG and AGC **(15 marks)**

1.

		0	1 A	2 A	3 T	4 G
0		0	-5	-10	-15	-20
1	A	-5	2	-3	-8	-13
2	G	-10	-3	-3	-8	-6
3	C	-15	-8	-8	-8	-11

$$F_{i,j} = \max \begin{cases} F_{i-1,j} + d & \uparrow \\ F_{i,j-1} + d & \leftarrow \\ F_{i-1,j-1} + S_{x_i y_j} & \nwarrow \end{cases}$$

1) The path (3,4), (2,4), (1,3), (1,2), (1,1), (0,0)

Optimal global alignment:

A	A	T	G	_
A	_	_	G	C

2) The path (3,4), (2,4), (1,3) (1,2), (0,1), (0,0).

optimal global alignment:

A	A	T	G	_
_	A	_	G	C

2.

		A	A	T	G
	0	0	0	0	0
A	0	2	2	0	0
G	0	0	0	0	2
C	0	0	0	0	0

$$F_{ij} = \max \begin{cases} F_{i-1,j} + d & \uparrow \\ F_{i,j-1} + d & \leftarrow \\ F_{i-1,j-1} + S_{ij} & \nwarrow \\ 0 \end{cases}$$

optimal local alignment:

A	T	G
A	_	G

Path: (2,4), (1,3), (1,2)

Question 5 (22 marks)

1. Define the class P in terms of computational complexity. Explain why problems in P are considered efficient. **(6 marks)**

2. Describe the class NP and provide an example of a problem that belongs to NP but not necessarily to P. Explain why solving such problems is more challenging. **(5 marks)**

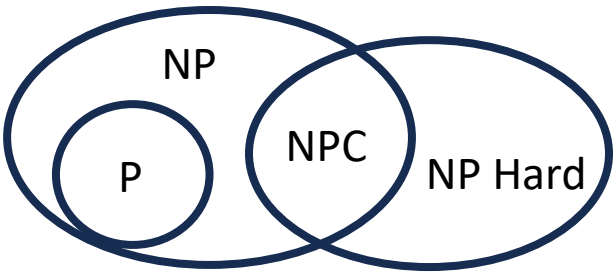
3. Define NP-complete (NPC) problems. Explain the significance of an NP-complete problem in the context of computational complexity theory. **(6 marks)**

4. Discuss the concept of polynomial-time reduction in the context of NP-completeness. How is it used to establish the NP-completeness of a problem? **(5 marks)**

1. Problems in class P means that they can be solved (by a turning machine) in polynomial time. It means that the computational complexity of them is less or equal than $O(n^k)(k \in \mathrm{Z}^+)$. As the problem size grows, the increase in computation time is manageable and not exponential, which is feasible to run on computers for real-world inputs. So they are considered efficient.
2. Class NP(Nondeterministic Polynomial time) problems represent a class of problems that can verify the correctness of answers in polynomial time. The Boolean Satisfiability Problem (SAT) is a problem that belongs to NP but not necessarily to P.
3. Class NPC(Non-deterministic Polynomial Complete) problems, if any algorithm of them exists in polynomial time, then all NP problems are polynomial time solvable. It is **significant** because if any one of them can be solved in polynomial time, then all problems in NP can be solved in polynomial time, causing P=NP.
4. Polynomial Time Reduction in NP Completeness refers to the transformation of an instance of a problem into an instance of another problem, which can be completed in polynomial time.

Class NP problems → reduce(via Polynomial Time Reduction) → Class NPC problems

If NP≠P:



Have a nice day! ^^