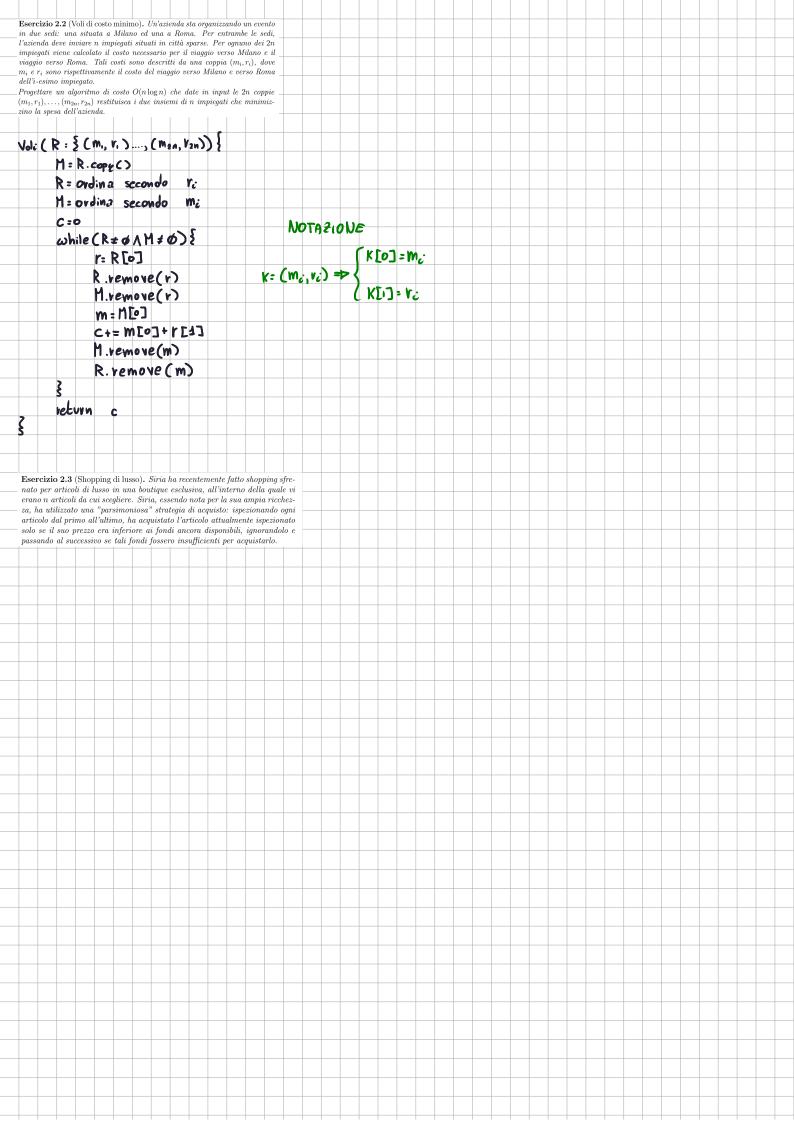
	POV	i 2 mc		he	ouni	node	, h	2 .	grado	div	2V2O.	ove	ini an	,	ali	Noc	); m	el	MC	odo	5	eave	nte	:	
1																									
									s(Vi)>																
h a	2	K'>	Κ≥	1	adi a	cent	i, 6	e'	chiavo	C	he	il	No	do	1/n	av	va′	ALY	ENO	n		adi a	? cen	Ŀi,	V
si		sono		al	Pi	υ' γ	<b>1-1</b>	noo	li ch	e	p055	on o	ess	ergl	: 6	zdi ac	enti	=>	Co	NTR	ADI	1210	NE		
		10/						0.16																	
rafo	com	plemer	tare de	G se p	er ogni e	arco(u, v)	) si veri	fica che	$iiamo \ \overline{G} \ co$ $(u,v) \in E(\overline{G} \ \grave{e} \ connes$	G)															
-				-1																					
) J	ppo	wi am	0	che	e	ntrai	иЬі	1	grafi	No	)M	Si	ano	CON	ness										
+																									
100	oia:	19/	Trode	minir	<u>n</u> ) De	moetra	che ar-	arafa	n diretto G																
n $cv$	ıi ogr	ni nodo	ha gra	do mag	giore o	uguale a			n airetto G re connesso.																
aff	erma	zione i	ate and	ene se C	F è dirett	to š																			
																	+++	-							

Esercizio 1.4 (Arcipelago). Un arcipelago è rappresentato da una matrice  $n \times m$ , dove ogni cella è marcata da uno 0, rappresentante il mare, o da un 1, rappresentante il terreno. In particolare, due celle appartengono alla stessa isola se e solo se sono marcate entrambe con 1 e sono adiacenti. Data in input la matrice M, progettare un algoritmo che in tempo O(nm) restituisca il numero di isole nell'arcipelago. Modificare l'algoritmo affinché restituisca la dimensione dell'isola di grandezza maggiore. Arcipelago (M[n.m]: matrice) { adj-island (M[n.m]:matrice, x:intero, y:intero) } For (i:0..., n) { \(\theta(n)\) For(i=x-1...,x+1) { \(\theta(3)\)} For (J:0..., m) { (m) For (7: 4-1..., 4+1) { 0(3) if (M[c,J] == 1){ c: clamp(0, n-1)  $K = ad_{J-island}(M, i, j) / \Theta(9)$ J = clamp (0, M-1) K:max(K,M[i,J]) if(K==1) { M[c, v] : C Belse & M[c, v] : K } return ni = 0 For(i:0...,n){ For (J=0..., m) { no=max(no, M[i,J] return ni-1 Biggest island (M [n.m]: matrice) { ni=Arcipelayo(H)// la funzione modifica la matrice A[ni+1]= {0,0...,0} For (i=0...,n-1) { For(7=0..., m-1) { iF(M[c,J] +0) { A[M[c,J]]++} bi=0 in:intero For (i = 0 ..., ni+1) { if(A[i]>bi){ bi=A[i] in=i return



	isca	la lungi	mo di co hezza il	cammir	no cre	escen	ite pi	iù lu	ngo d	al suo	interr	ю.	w 01		ماد	ء ال	100	atı	iica		e'	UN	noc	10	e	4	ha	n	
															alov												7,0		
2vco		erso			di		Ü	O	2361	a) (	310	0		700	CIOV	2	ma	יכני	ove										
		atric 3v a F	e n	(m×																									F
		(i:1	n){	,																									
		Fo	v(J:1	,m)	<b>{</b>																								
		2	V	(G).	add	(	Ci,	1))																					
	3																												F
		(i:1	,n){																										
		Fo	v(J=1	,m)	1		ME	•.•	2/	5																			L
			j (	=C M[	c,j.	) )<	unr	(	1] [2.5]	). (/	+1.3	9																	H
			3																										Ĺ
			į f	=C M[	6,3	) د	ME	,3+1	))	٤																			L
			3		ECG	). 8	3 <b>od</b>	( (	i,t	),(i	1341	עו											+						
		}	,																										
	3			h.	, ,			. 1								•													L
	reŁ	UVN	m az	x_di!	stC	G	) //	di	st a	ry 20	9	me	)55i	M2	F		2		V Oc	i									H
																													r
																													Ĺ
																													L
																													H
																													I
																													-
																													H
																													I
																							-						F
																													H
																							_						L
																													H
																													L
																													H
																													İ
																													Ĺ
					1						1																		
																													t



Esercizio 2.4 (Ciclo di peso minimo). Dato un grafo con pesi positivi G, il Per trovare un ciclo, faccio una DFS, quando considero un nodo che si trova ancora nello Stack, allora vuol dire che vi è un ciclo. peso di un ciclo all'interno di G è la somma dei pesi degli archi del ciclo.  $Progettare\ un\ algoritmo\ che\ dato\ in\ input\ il\ grafo\ G\ restituisca\ l'insieme\ di$ archi che compone il ciclo di peso minimo presente in G (restituire  $\varnothing$  se Gè aciclico). La complessità prevista dell'algoritmo è  $O(m(n+m)\log n)$ . Compute\_cycle (G:grafo, 5:stack, u:nodo) { // questa funzione considerato vestituisce cicle 55 = 5.copy () // mi prondo copia A={ } // conterva gli archi V = U do { ω = 53.pop() A.add( (ω,4)) 1 = W } while (w = u) return A MinCecle (G: groFo) { Vis[n]= {0,0...,0} // array lungo n ins[n]={0,0...,0} // aveny lungo n, se in5[4]:1 ←> 4 e' 5:Stack Out = } outW=00 For each usv(c) { //nel sia diretto/non connesso caso il grofo iF (Vis[4]::0) { Vis[4]:1 S.push (U) in5[4]=1 while (5 + Ø) { x = 5. top () ex:0 For each ye = adj() { iF(Vis[Y] # 1){ ex=1 S. push(x) Dist[2]=1 inS[4]=1 if (ex == 0) } //x va bolto dallo stack For each yex. adj() { if (in S[4] == 1) { // controllo se Forma C = Compute\_Cycle (G, S, Y) IF (W(C) < OULW) { outy = w(c) Out = C in [ ]. pop [ ] = 0 return



Esercizio 4.1 (Sotto-array di prodotto massimo). Dato un array A di n	
interi (positivi, nulli o negativi), progettare un algoritmo di complessità O(n)	
che restituisca il prodotto massimo ottenibile con gli elementi di un sotto- array (un sotto-array contiene elementi contigui di A).	
Considero T[i]= Prodolto massimo di un solto avvaz di A[o:i]	contenente A[i]
PoMase (A: aveay) {	
n=A.length()	
T[n]={o,o,o}	
Te]==[e]	
For (c= 1,2n-1)	
T[i]= max(A[i], T[i-1]-A[i])	
m: max (m, T[i])	
iF(m40) {veturn o}	
return m	
\$	
Esercizio 4.2 (Numero di passeggiate). Progettare un algoritmo che dato	
in input un grafo diretto $G$ , due vertici $x,y\in V(G)$ e un valore $k\in\mathbb{N}$	
restituisca il numero di passeggiate da x a y distinte di lunghezza al massimo k. La complessità dell'algoritmo deve essere O(nmk)	
NumPass (G:grafo, x:nodo, x:nodo, K:intero) }	
T[nxk]:matrise	
T[x,0]:1	
Fov (c = 0,1 n-1) {	
$  F(i \in \mathbb{Z}, 205) \{ T[i, 4] = 1 \} $ $  e se \{ T[i, 4] = 0 \} $	
}	
For (i = 0, 4, M-1) {	
Fov (3 = 1, 2 K) {	
For each (u e i. ads) {	
5+=T[u.5-i]	
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \	
50)=0	
For(i=0K) {	
Sol+=T[x, i]	
<u> </u>	
vetum sol	
<u> </u>	

