Contents

1	Geo	metry	-																				2
	1.1	一些公式																					2
		1.1.1	Her	on's	Fo	rm	ıula	١.															2
		1.1.2	四面			-																	2
		1.1.3 1.1.4	三角三角													٠	•			•	•	٠	2
		1.1.5	二角																				2
		1.1.6	三角																				2
		1.1.7	三角																				2
		1.1.8 1.1.9	Pick Eule																	•	•	٠	2
	1.2	1.1.9 三角公式	Eure	:15	1.01	m		少 1:												•	•	•	2
		1.2.1	超球	坐材	下系																		2
		1.2.2	三维			式																	2
		1.2.3	立体																		•	٠	2
		1.2.4 1.2.5	常用扇形																		•	•	2
		1.2.6	高维																				2
	1.3	距离																					2
	1.4	Pick 定理																					2
	1.5 1.6	二维计算, 三角形,	儿19 3					٠.								٠				•	•	٠	3
	1.7	二用ル · 凸包																					3
	1.8	半平面交																					4
2	Gra	ph																					5
	2.1	图论基本:	知识																				5
		2.1.1	树链																				5
		2.1.2 2.1.3	帯修															•		•	•	٠	5 5
		2.1.3	差分			•												•		•	•	•	5
		2.1.5	Segi			ree																	5
		2.1.6	二分	_	٠.																		5
		2.1.7	稳定																		•		5
		2.1.8 2.1.9	三元图同																		•	٠	5 5
																					•	•	5
		2.1.10	竞赛	. 图 I	∠an	aaı	ı s	Ine	105	en	n.												J
		2.1.11	竞赛 Ran	isey	Th	eo	ren	ı R((3,3)	3)=	6,	R(4	1,4)	=1	8								5
		2.1.11 2.1.12	Ran 树的	isey 计数	Th t P	eo: ruf	ren er月	n R(序列	(3,3	3)=	6,	R(4	1,4) 	=1	8								5
		2.1.11 2.1.12 2.1.13	Ran 树的 有根	nsey 计数 树的	Th 女 P	ieo ruf 数	ren er月	n R(序列 	(3,3	3)=	6,	R(4	l,4))=1 ·	. 8					· ·			5 5 5
		2.1.11 2.1.12	Ran 树的 有根 无根	nsey 计数 树的	Th 及P 的计	eo ruf 数 数	ren er店 · ·	i R(序列 	(3,3 · ·	3)=	6,	R(4	l,4) · ·)=1	. 8								5
		2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16	Ram 树有无生有	nsey 计树树树图	Then的计数拉	rufe y 数数 Ki 回	ren er /i · · · rch 洛计	n R(序列 · · · hof	(3,3 BE	3)=	6, latı	R(4	1,4) Tr ere)=1 ee m			er	en	 n .				5 5 5 5 5 5
		2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17	Ram 村有无生有 Tutt	nsey 计树树树图 e M	Th 文 为 力 力 大 数 拉 tatr	rufo y 数数 Ki 回 ix	ren er月 · · · rch 恪计	n R(序列 · · · hof	(3,3 f's BE	3)=	6, 1	R(4	1,4) · · · · Tr ere)=1 ee m	.8 TI		er	en	 n .	· ·			5 5 5 5 5 5 5
		2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18	Ram 树有无生有Tutt Edm	nsey 计树树树图 图 M none	Th 対 対 対 対 対 数 対 は atr	rufo y 数数 Ki 回 ix Mat	ren er /s rch 答计	n R(序列 · · · hof 数	(3,3 f's BE	3)=	6, i	R(4	1,4))=] ee m	.8 		er	en	 n .				5 5 5 5 5 5 5 5
	2.2	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17	Ram 的根根成向tt Edm 向	nsey 计树树树图 图 M none	Th 为 为 为 力 大 d s d s d s 和 s 和 s a tr d s a tr d a tr d a tr d a tr d a tr d a tr d a a a a a a a a a a a a a	rufe rufe 数数Ki 回ix Mat	remer sers sers sers sers sers sers sers	n R(序列 · · · hof 数	(3,5 f's BE	3)= · · · · · · · · · · · · · · · · · · ·	6, latı	R(4	1,4) · · · · Tr ere	:=1 eee m	8 TI		er	en					5 5 5 5 5 5 5
	2.3	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19	Ram 的根根成向tt Edm 向	nsey 村树树图 e M none 	Th 为 为 为 力 大 d s d s d s 和 s 和 s a tr d s a tr d a tr d a tr d a tr d a tr d a tr d a a a a a a a a a a a a a	rufo rufo 数数Ki 回ix Mat Mat	remer / · · · · · · · · · · · · · · · · · ·	n R(序列 	(3,: · · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6, 1	R(4	1,4) 	:=1 	.8 			en					5 5 5 5 5 5 5 5 5
	2.3 2.4	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT . 极大团 . k短路	Ram的根根成向tt Edn · · ·	nsey 计树树树图 Manono	· Thy Apply the Apply that the Apply the Appl	rufo rufo 数数Ki: 以at Mat	rem er序 · · · rch 答计 · · crix 向, 1	n R(F列	(3,: f's BE · · 项	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	:=] 	.8 TI			en					5 5 5 5 5 5 5 5 5 5 6 6
	2.3 2.4 2.5	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT . k短路	Ram 的根根成向tt Edn Control	nsey 対 材 材 材 内 N N N N N N N N N N N N N	· Thy P · Y P · Y D · H · H · H · H · H · H · H · H	rufe rufe 数数 Ki 回 ix Mat · · · ·	rem er 序 · · · · rch · · · · · · · · · · · · · · ·	n R(F列 hot 数 色	(3,5 · · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	.8 .TI								55 55 55 55 55 55 56 66 77
	2.3 2.4 2.5 2.6	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT 极大路	Ram的根根成向tt Edn · · · · ·	usey 树树树图 e M none A	· The P的 计数位 I atri	rufe rufe 数数 Kii 回 ix Mat Mat	rem er 序 · · · · rrch * · · · · · · · · · · · · · · · · · · ·	n R(序列	(3,3,5 · · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	.8 TII 								55 55 55 55 55 55 56 66 77
	2.3 2.4 2.5	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT . k短路	Ram的根根成向tt Edn · · · · ·	usey 树树树图 e M none A	· The P的 计数位 I atri	rufe rufe 数数 Kii 回 ix Mat Mat	rem er 序 · · · · rrch * · · · · · · · · · · · · · · · · · · ·	n R(F列 hot 数 色	(3,3,5 · · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	.8 TII 								55 55 55 55 55 55 56 66 77
3	2.3 2.4 2.5 2.6 2.7	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · 极大团 · k短路 · . KM · . · . 最小斯坦:	Ram 树有无生有Tuth Edm 向····树	usey 對的 對的 的 one one one one one one one one	· The P的 计数位 I atri	rufe rufe 数数 Kii 回 ix Mat Mat	rem er 序 · · · · rrch * · · · · · · · · · · · · · · · · · · ·	n R(序列	(3,3,5 · · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	.8 TII 								55 55 55 55 55 55 56 66 77
3	2.3 2.4 2.5 2.6 2.7 Dat	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT 极大团 · k短路 · · · · · · · · · · · · · · · · · · ·	Ram 树有无生有Tutt Edm 向····树 t u	usey 数 数 数 数 数 的 的 的 的 的 的 的 的 的 的 的 的 的	· The 分子 The Company of the Company	rufo rufo 数 Kii jix Mat iix Mat	remerserserserserserserserserserserserserse	n R((3,5) f's BE · · · 项 · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 66 67 77 88
3	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · 极大团 · k短路 · · · KM · · · · tarjan · · · · · · · · · · · · · · · · · · ·	Ram 树有无生有Tutt Edm · · · · · 树 Ct树	usey 数 数 数 的 的 的 的 的 的 的 的 的 的 的 的 的	· The Phih 数拉tats A · · · · · · · · · · · · · · · · · ·	leof ru 数数Kii Ju Mat ix Mat ix	remer fer fer fer fer fer fer fer fer fer f	n R(F) No. 1 No.	(3,5) f's BE · · 项 · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 55 66 67 77 88
3	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · · · · · · · · · · · · · · · · · · ·	Ram 树有无生有Tutt Edn · · · · · 树 · · · 树 · · · · · · 树 · · · · · · · · · · · · 树 ·	nsey 数 的 的 的 的 的 的 的 的 的 的 的 的 的	· Tr P 计计数拉trist in the state of the state	leof ruf 数数Kii Mat Mat ii	remer fer fer fer fer fer fer fer fer fer f	n R(F) No hot M M M M M M M M M M M M M M M M M M M	(3,5) F's BE · · 项 · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6, 1	R(4	1,4)	ee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 66 67 77 88
3	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT k短短 · · · · · · · · · · · · · · · · · ·	Ram的根根成向tttE有····纳 Ct树、树段对下Edm向····树 U	nsey 数的 数的 的 数的 的 的 的 的 的 的 的 的 的 的 的 的 的	· The Phity Material Services And Services	leof ruf 数数Kii Mat Mat ii	remer/ser/ser/ser/ser/ser/ser/ser/ser/ser/s	n R(n R(n R(n R(n R(n R(n R(n R(n R(n R((3,5° sf's BE · · 项 · · · · · · · · · · · · · · · ·	3)= 	6, 1	R(4	1,4)	ee m · · · · · · · · · · · · · · · · · ·	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 56 66 77 78 89 99 99
3	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · k短路 · · · · k短路 · · · · tarjan · · · · · · · · · · · · · · · · · · ·	Ram的根根成向tttE有····纳 Ct树、树段对下Edm向····树 U	nsey 数时 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的	The Phith 数拉trians the state of the state o	ruff ruff 数数Kii Jix Mat li Li Li Li Li Li Li Li Li Li Li Li Li Li	ren er 序 · · · · · · · · · · · · · · · · · · ·	n R() R() N() N() N() N() N() N() N() N() N() N	(3,;; fr)'s BE · · · 项 · · · · · · · · · · · · · · ·	3)= 	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 66 67 77 88
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · 极大路 · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttE有····纳 Ct树,树段ep	nsey 数时 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的	The Phith 数拉trians the state of the state o	ruff ruff 数数Kii Jix Mat li Li Li Li Li Li Li Li Li Li Li Li Li Li	ren er 序 · · · · · · · · · · · · · · · · · · ·	n R() R() N() N() N() N() N() N() N() N() N() N	(3,;; fr)'s BE · · · 项 · · · · · · · · · · · · · · ·	3)= 	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 56 66 77 78 8 9 9 9 9 10 11 11 11
3	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · 极大路 · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttE有····纳 Ct树,树段ep	nsey 数时 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的 的	The Phith 数拉trians the state of the state o	ruff ruff 数数Kii Jix Mat li Li Li Li Li Li Li Li Li Li Li Li Li Li	ren er 序 · · · · · · · · · · · · · · · · · · ·	n R() R() N() N() N() N() N() N() N() N() N() N	(3,;; fr)'s BE · · · 项 · · · · · · · · · · · · · · ·	3)= 	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 56 66 77 78 8 9 9 9 9 10 11 11 11
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttdm向·····树 tdM·树段ep········树 td····树·····树·········村 td·······村 ···村··············	nsey 当时 新的 所的 所的 所的 所的 所的 所的 所的 所的 所的 所	The P. 计计数拉tri A. The P. 计计数拉tri A. The P. 计计数拉tri A. The P. The	rufo rufo 数数Kib Matri	ren er /s rch 格····································	n R(n R) · · · · · · · · · · · · · · · · · ·	(3, : f's BE 项	3)= · · · · M SST · · · 式 · · · · · · · · · · · · · · ·	6,	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 56 66 77 77 88 99 99 100 111 112 122 122
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri. 4.1 4.2	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · KM · · · · センスを表する。 KM · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttm	nsey 数计树树树图 e Mono no	, 放的分十数Later · · · · · · · · · · · · · · · · · · ·	rufe rufe 数数Kii jix Mati	ren er /s rch 格····································	n R(R)	(3, ; · · · · · · · · · · · · · · · · · ·	3)= 	6,	R(4	11,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 55 66 67 77 88 99 99 100 111 112 122 122 122
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri 4.1 4.2 4.3	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 极大团 k短短点, KM	Ram的根根成向ttm向··································	nsey 数计树树树图 Mono Re M	, 放射的计数 Later Lat	rufe rufe 数数Kii jix Mate	ren er // crix 句,1	n R((3,5 · · · f's BE · · 项 · · · · · · · · · · · · · · · ·	3)=	6, (atr	R(4	1,4)	eee m	8 · · · · · · · · · · · · · · · · · · ·			en					55 55 55 55 55 55 55 55 55 66 67 77 88 99 99 101 111 112 122 122 122 122 122 122 123 124 125 125 125 125 125 125 125 125 125 125
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri. 4.1 4.2	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · KM · · · · センスを表する。 KM · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttm向··································	nsey 数计树树树图 Mono Mono Mono Mono Mono Mono Mono Mono	The Phylogen Harden Ha	rufo rufo 数数Kii Juate Mate	ren er // crix 句,1	n R((3, 5 · · · · · · · · · · · · · · · · · ·	3)= · · · MSS· · · 式 · · · · · · · · · · · · · · ·	6, (atr	R(4	1,4)	ee m	8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 55 66 67 77 88 99 99 100 111 112 122 122 122
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri. 4.1 4.2 4.3 4.4 4.5 4.6	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT · · · · · · · · · · · · · · · · · · ·	Ram的根根成向ttransin de t 树 树 be de the state of the state o	nsey 数计树树树图 en图 · · · · · · · · · · · · · · · · · · ·	The Phih 数粒tri State St	rufo rufo 数 Ki B ix Mati	renger A	n R(R)	(3, 5 · · · · · · · · · · · · · · · · · ·	3)= · · · · · · · · · · · · · · · · · · ·	6, 1	R(#	1,4)	eee m	8 8								55 55 55 55 55 55 55 55 55 55 56 66 77 77 88 99 99 100 111 112 122 122 122 121 121 121 121
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri 4.1 4.2 4.3 4.4 4.5 4.6 4.7	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT k短短 KM 坦 2 SAT KD Tree 李吉HQ Tree 卷 表自树 Manache 字 SA SAM	Ram的根根成向ttdr向 · · · · · 纳 Ct树 · 树段ep · · · · · · · · · · · · · · · · · · ·	nsey 数计树树树图 e Mono nono re	The Phih 数粒tri Andrews Andrew	rufd rufd 数Kib Math · · · · · · · · · · · · · · · · · · ·	renger per per per per per per per per per p	n R(R	(3,:	3)= · · · · · · · · · · · · · · · · · · ·	66, 1	R(4	1,4)		8 8								55 55 55 55 55 55 55 55 55 56 66 77 77 88 99 99 100 111 112 122 122 122 123 133 133 133 133
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT LET WE KM LATION TO WE	Ram的根根成向ttdr向 · · · · · · · · · · · · · · · · · · ·	nsey 数计树树树图 e Mono Nono Nono Nono Nono Nono Nono Nono	· The Phith教拉tri State	rufe rufe 数K回ix M定····································	renger she is a series of the crix of the	n R(R)	(3,:	3))= · · · · · · · · · · · · · · · · · ·	66, 1	R(#	1,4)		8 · · · · · · · · · · · · · · · · · · ·								55 55 55 55 55 55 55 55 55 66 67 77 88 99 100 111 112 122 122 122 123 133 133 133 133
	2.3 2.4 2.5 2.6 2.7 Dat 3.1 3.2 3.3 3.4 3.5 3.6 Stri 4.1 4.2 4.3 4.4 4.5 4.6 4.7	2.1.11 2.1.12 2.1.13 2.1.14 2.1.15 2.1.16 2.1.17 2.1.18 2.1.19 2 SAT k短短 KM	Ram的根根成向ttraf · · · · · 纳 · ct 树 · 树段ep · · · · · · · · · · · · · · · · · · ·	usey 数 対 材 材 材 内 の の の の 、 、 、 、 、 、 、 、 、 、 、 、 、	The Phith 数粒 tate reserved to the second se	rufd ix M定····································	renger she is the state of the	n R(R)	(3,5 · · · · · · · · · · · · · · · · · · ·	3)= · · · · MSST · · · 式 · · · · · · · · · · · · · · ·	66, 1	R(4	1,4)		8 8								55 55 55 55 55 55 55 55 55 56 66 77 77 88 99 99 100 111 112 122 122 122 123 133 133 133 133

1. Geometry

1.1 一些公式

1.1.1 Heron's Formula

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

$$p = \frac{a+b+c}{2}$$

1.1.2 四面体内接球球心 假设 s_i 是第 i 个顶点相对面的面积,则

$$\begin{cases} x &= \frac{s_1x_1 + s_2x_2 + s_3x_3 + s_4x_4}{s_1 + s_2 + s_3 + s_4} \\ y &= \frac{s_1y_1 + s_2y_2 + s_3y_3 + s_4y_4}{s_1 + s_2 + s_3 + s_4} \\ z &= \frac{s_1z_1 + s_2z_2 + s_3z_3 + s_4z_4}{s_1 + s_2 + s_3 + s_4} \end{cases}$$

体积可以使用 1/6 混合积求, 内接球半径为

$$r = \frac{3V}{s_1 + s_2 + s_3 + s_4}$$

1.1.3 三角形内心

$$\vec{I} = \frac{a\vec{A} + b\vec{B} + c\vec{C}}{a + b + c}$$

1.1.4 三角形外心

$$\vec{O} = \frac{\vec{A} + \vec{B} - \frac{\overrightarrow{BC} \cdot \overrightarrow{CA}}{\overrightarrow{AB} \times \overrightarrow{BC}} \overrightarrow{AB}^T}{2}$$

1.1.5 三角形垂心

$$\vec{H} = 3\vec{G} - 2\vec{O}$$

1.1.6 三角形偏心 $-a\vec{A} + b\vec{B} + c\vec{C}$

交点,外切圆圆心.剩余两点的同理. 1.1.7 三角形内接外接圆半径

$$r = \frac{2S}{a+b+c}$$
, $R = \frac{abc}{4S}$

1.1.8 Pick's Theorem 格点多边形面积

 $S = I + \frac{B}{2} - 1$. I 内部点, B 边界点。

1.1.9 Euler's Formula 多面体与平面图的点、边、面

For convex polyhedron: V - E + F = 2.

For planar graph: |F| = |E| - |V| + n + 1, n : #(connected components).

1.2 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$

$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a) \tan(b)}$$

$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a) \cos(b)}$$

$$\sin(a) + \sin(b) = 2\sin(\frac{a + b}{2})\cos(\frac{a - b}{2})$$

$$\sin(a) - \sin(b) = 2\cos(\frac{a + b}{2})\sin(\frac{a - b}{2})$$

$$\cos(a) + \cos(b) = 2\cos(\frac{a + b}{2})\cos(\frac{a - b}{2})$$

$$\cos(a) - \cos(b) = -2\sin(\frac{a + b}{2})\sin(\frac{a - b}{2})$$

 $\sin(na) = n\cos^{n-1} a \sin a - \binom{n}{3}\cos^{n-3} a \sin^3 a + \binom{n}{5}\cos^{n-5} a \sin^5 a - \dots$

$$\cos(na) = \cos^n a - \binom{n}{2} \cos^{n-2} a \sin^2 a + \binom{n}{4} \cos^{n-4} a \sin^4 a - \dots$$

1.2.1 超球坐标系

$$\begin{array}{rcl} x_1 & = & r\cos(\phi_1) \\ x_2 & = & r\sin(\phi_1)\cos(\phi_2) \\ \dots & & \\ x_{n-1} & = & r\sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n & = & r\sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1}) \\ \phi_{n-1} & \in & [0,2\pi] \\ \forall i = 1..n - 1\phi_i & \in & [0,\pi] \end{array}$$

1.2.2 三维旋转公式

绕着 (0,0,0)-(ux,uy,uz) 旋转 θ , (ux,uy,uz) 是单位向量

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1.2.3 立体角公式

$$\phi$$
: 二面角
$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac}) \text{ rad} - \pi \text{ sr}$$

$$\tan\left(\frac{1}{2}\Omega/\mathrm{rad}\right) = \frac{\left|\vec{a}\ \vec{b}\ \vec{c}\right|}{abc + \left(\vec{a}\cdot\vec{b}\right)c + \left(\vec{a}\cdot\vec{c}\right)b + \left(\vec{b}\cdot\vec{c}\right)a}$$

$$\theta_s = \frac{\theta_a + \theta_b + \theta_c}{2}$$

1.2.4 常用体积公式

- 棱锥 Pyramid $V = \frac{1}{3}Sh$.
- \Re Sphere $V = \frac{4}{3}\pi R^3$.
- $\oint Frustum V = \frac{1}{3}h(S_1 + \sqrt{S_1S_2} + S_2).$
- 椭球 Ellipsoid $V = \frac{4}{3}\pi abc$.
- $\pi \oplus Spherical cap \frac{\pi}{3}(3R-H)H^2$

1.2.5 扇形与圆弧重心

扇形重心与圆心距离为 $\frac{4r\sin(\theta/2)}{3\theta}$, 圆弧重心与圆心距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

1.2.6 高维球体积

$$\begin{split} V_2 &= \pi R^2, \, S_2 = 2\pi R \\ V_3 &= \frac{4}{3}\pi R^3, \, S_3 = 4\pi R^2 \\ V_4 &= \frac{1}{2}\pi^2 R^4, \, S_4 = 2\pi^2 R^3 \\ \text{Generally,} \, V_n &= \frac{2\pi}{n} V_{n-2}, \, S_{n-1} = \frac{2\pi}{n-2} S_{n-3} \\ \text{Where,} \, S_0 &= 2, \, V_1 = 2, \, S_1 = 2\pi, \, V_2 = \pi \end{split}$$

1.3 距离

欧式距离

$$\sqrt{\sum_{i=1}^{n} (x_{1,i} - x_{2,i})^2}$$

曼哈顿距离

$$\sum_{i=1}^{n} |x_{1,i} - x_{2,i}|$$

切比雪夫距离

$$\max_{i=1}^{n} \{ |x_{1,i} - x_{2,i}| \}$$

曼哈顿距离与切比雪夫距离转换:

- 曼哈顿坐标系是通过切比雪夫坐标系旋转 45°后,再缩小到原来的一半得到的。
- 将一个点(x,y)的坐标变为(x+y,x-y)后,原坐标系中的曼哈顿距离等于新坐标系中的切比雪夫距离。
- 将一个点 (x,y) 的坐标变为 $(\frac{x+y}{2},\frac{x-y}{2})$ 后,原坐标系中的切比雪夫距离 等于新坐标系中的曼哈顿距离。

1.4 Pick 定理

给定顶点均为整点的简单多边形,皮克定理说明了其面积 A 和内部格点数目 i、边上格点数目 b 的关系:

$$A = i + \frac{b}{2} - 1$$

推广:

- 取格点的组成图形的面积为一单位。在平行四边形格点,皮克定理依然成立。套用于任意三角形格点,皮克定理则是 $A=2\times i+b-2$ 。
- 对于非简单的多边形 P, 皮克定理 A = i + b/2 χ(P), 其中 χ(P) 表示 P
 的欧拉特征数 χ(P) = V E + F。
- 皮克定理和欧拉公式 (V-E+F=2) 等价。

1.5 二维计算几何基础

```
#define cp const vec &
   #define cl const line &
 3
   struct vec {
       vec rot(db t) const { // 逆时针
           return \{x * \cos(t) - y * \sin(t), x * \sin(t) + y *
              vec rot90() const { return {-y, x}; }
       db len2() const { return x * x + y * y;}
db len() const { return sqrt(x * x + y * y);}
 8
       vec unit() const { db d = len(); return \{x / d, y / d\};
10
   };
   struct line { vec s, t; }
11
   bool turn_left(cp a, cp b, cp c) {
   | return sgn (crs (b - a, c - a)) >= 0; }
   bool point_on_segment(cp a, cl b) { // 点在线段上
14
       return sgn(crs(a - b.s, b.t - b.s)) == 0 // 在直线上
              && sgn(dot(b.s - a, b.t - a)) <= 0; }
```

```
17 bool two_side(cp a, cp b, cl c) {
       return sgn(crs(a - c.s, c.t - c.s))
18
19
               * sgn(crs(b - c.s, c.t - c.s)) < 0; }
   bool intersect_judge(cl a, cl b) { // 线段判非严格交
20
21
       if (point_on_segment(b.s, a)
           || point_on_segment(b.t, a)) return true;
23
       if (point_on_segment(a.s, b)
24
           || point_on_segment(a.t, b)) return true;
25
       return two_side(a.s, a.t, b)
26
              && two_side(b.s, b.t, a); }
   vec line_intersect(cl a, cl b) { // 直线交点
27
28
       db \ s1 = crs(a.t - a.s, b.s - a.s);
       db \ s2 = crs(a.t - a.s, b.t - a.s);
29
       return (b.s * s2 - b.t * s1) / (s2 - s1); }
30
31
   bool point_on_ray(cp a, cl b) { // 点在射线上
       return sgn(crs(a - b.s, b.t - b.s)) == 0
32
33
              && sgn(dot(a - b.s, b.t - b.s)) >= 0; }
34
   bool ray_intersect_judge(line a, line b) { // 射线判交
       db s1, s2; // can be LL
35
36
       s1 = crs(a.t - a.s, b.s - a.s);
37
       s2 = crs(a.t - a.s, b.t - a.s);
38
       if (sgn(s1) == 0 \&\& sgn(s2) == 0) {
39
           return sgn(dot(a.t - a.s, b.s - a.s)) >= 0
40
                   || sgn(dot(b.t - b.s, a.s - b.s)) >= 0; }
       if (!sgn(s1 - s2) || sgn(s1) == sgn(s2 - s1)) return 0;
41
42
       swap(a, b);
43
       s1 = crs(a.t - a.s, b.s - a.s);
       s2 = crs(a.t - a.s, b.t - a.s);
       return sgn(s1) != sgn(s2 - s1); }
45
   db point_to_line(cp a, cl b) { // 点到直线距离
46
       return abs(crs(b.t - b.s, a - b.s)) / dis(b.s, b.t); }
47
48
   vec project_to_line(cp a, cl b) { // 点在直线投影
49
       return b.s + (b.t - b.s)
              * (dot(a - b.s, b.t - b.s) / (b.t -
50
                db point_to_segment(cp a, cl b) { // 点到线段距离
51
52
       if (sgn(dot(b.s - a, b.t - b.s))
            sgn(dot(b.t - a, b.t - b.s)) <= 0)
53
54
           return abs(crs(b.t - b.s, a - b.s)) / dis(b.s,
55
       return min(dis(a, b.s), dis(a, b.t)); }
56
   bool in_polygon(cp p, const vector <vec> &po) {
       int n = (int) po.size(); int cnt = 0;
57
58
       for (int i = 0; i < n; ++i) {
59
           vec a = po[i], b = po[(i + 1) \% n];
           if (point_on_segment(p, line(a, b))) return true;
60
61
           int x = sgn(crs(p - a, b - a)),
62
               y = sgn(a.y - p.y), z = sgn(b.y - p.y);
           if (x > 0 \&\& y \le 0 \&\& z > 0) ++cnt;
63
64
           if (x < 0 \&\& z <= 0 \&\& y > 0) --cnt; }
       return cnt != 0; }
65
   vector <vec> line_circle_intersect(cl a, cc b) {
       if (sgn(point_to_line(b.c, a) - b.r) > 0)
67
68
           return vector <vec> ();
69
       db x = sqrt(sqr(b.r) - sqr(point_to_line(b.c, a)));
70
       return vector <vec>
               ({project_to_line(b.c, a) + (a.s - a.t).unit() *
71
72
                project_to_line(b.c, a) - (a.s - a.t).unit() *
                  \hookrightarrow x}); }
   db circle_intersect_area(cc a, cc b) {
73
       db d = dis(a.c, b.c);
       if (sgn(d - (a.r + b.r)) >= 0) return 0;
75
76
       if (sgn(d - abs(a.r - b.r)) \leftarrow 0) {
77
           db r = min(a.r, b.r);
           return r * r * PI; }
78
       db x = (d * d + a.r * a.r - b.r * b.r) / (2 * d),
79
80
              t1 = acos(min(1., max(-1., x / a.r))),
81
              t2 = acos(min(1., max(-1., (d - x) / b.r)));
       return sqr(a.r) * t1 + sqr(b.r) * t2 - d * a.r *
82
         \hookrightarrow sin(t1); }
   vector <vec> circle_intersect(cc a, cc b) {
       if (a.c == b.c
84
           || sgn(dis(a.c, b.c) - a.r - b.r) > 0
85
86
           || sgn(dis(a.c, b.c) - abs(a.r - b.r)) < 0)
           return {};
87
88
       vec r = (b.c - a.c).unit();
       db d = dis(a.c, b.c);
89
90
       db x = ((sqr(a.r) - sqr(b.r)) / d + d) / 2;
       db h = sqrt(sqr(a.r) - sqr(x));
91
       if (sgn(h) == 0) return \{a.c + r * x\};
92
       return {a.c + r *x + r.rot90() *h,
```

```
a.c + r *x - r.rot90() *h; }
   // 返回按照顺时针方向
95
   vector <vec> tangent(cp a, cc b) {
97
       circle p = make_circle(a, b.c);
98
       return circle_intersect(p, b); }
   vector <line> extangent(cc a, cc b) {
       vector <line> ret;
100
       if (sgn(dis(a.c, b.c) - abs(a.r - b.r)) \leftarrow 0) return

→ ret:
02
       if (sgn(a.r - b.r) == 0) {
103
            vec dir = b.c - a.c;
           dir = (dir * a.r / dir.len()).rot90();
104
           ret.push_back(line(a.c + dir, b.c + dir));
105
106
           ret.push_back(line(a.c - dir, b.c - dir));
107
       } else {
           vec p = (b.c * a.r - a.c * b.r) / (a.r - b.r);
109
           vector pp = tangent(p, a), qq = tangent(p, b);
           if (pp.size() == 2 && qq.size() == 2) {
                if (sgn(a.r - b.r) < 0)
                    swap(pp[0], pp[1]), swap(qq[0], qq[1]);
113
                ret.push_back(line(pp[0], qq[0]));
114
                ret.push_back(line(pp[1], qq[1])); } }
115
       return ret; }
116
   vector <line> intangent(cc a, cc b) {
117
       vector <line> ret;
       vec p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
118
119
       vector pp = tangent(p, a), qq = tangent(p, b);
20
       if (pp.size() == 2 && qq.size() == 2) {
           ret.push_back(line(pp[0], qq[0]));
           ret.push_back(line(pp[1], qq[1])); }
       return ret; }
124
   vector <vec> cut(const vector<vec> &c, line p) {
25
       vector <vec> ret;
       if (c.empty()) return ret;
126
27
       for (int i = 0; i < (int) c.size(); ++i) {
           int j = (i + 1) \% (int) c.size();
128
29
           if (turn_left(p.s, p.t, c[i])) ret.push_back(c[i]);
           if (two_side(c[i], c[j], p))
30
131
                ret.push_back(line_intersect(p, line(c[i],
                  return ret; }
132
```

1.6 三角形

```
vec incenter(cp a, cp b, cp c) { // 内心
       db p = dis(a, b) + dis(b, c) + dis(c, a);
       return (a * dis(b, c) + b * dis(c, a) + c * dis(a, b))
 3
         vec circumcenter(cp a, cp b, cp c) { // 外心
       vec p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2
         \hookrightarrow 2);
       db d = crs(p, q);
       return a + vec(crs(s, vec(p.y, q.y)), crs(vec(p.x,
         \hookrightarrow q.x), s)) / d; }
   vec orthocenter(cp a, cp b, cp c) { // 垂心
       return a + b + c - circumcenter(a, b, c) * 2.0; }
10
   vec fermat_point(cp a, cp b, cp c) { // 费马点
       if (a == b) return a;
       if (b == c) return b;
12
       if (c == a) return c;
       db ab = dis(a, b), bc = dis(b, c), ca = dis(c, a);
14
15
       db cosa = dot(b - a, c - a) / ab / ca;
       db cosb = dot(a - b, c - b) / ab / bc;
16
17
       db cosc = dot(b - c, a - c) / ca / bc;
18
       db sq3 = PI / 3.0; vec mid;
       if (sgn(cosa + 0.5) < 0) mid = a;
19
       else if (sgn(cosb + 0.5) < 0) mid = b;
20
21
       else if (sgn(cosc + 0.5) < 0) mid = c;
22
       else if (sgn(crs(b - a, c - a)) < 0)
           mid = line_intersect(line(a, b + (c - b).rot(sq3)),
              \hookrightarrow line(b, c + (a - c).rot(sq3)));
            mid = line_intersect(line(a, c + (b - c).rot(sq3)),
25
              \hookrightarrow line(c, b + (a - b).rot(sq3)));
26
       return mid; } // minimize(|A-x|+|B-x|+|C-x|)
```

1.7 凸包

```
vector<vec> convex_hull(vector<vec> a) {
   int n = (int) a.size(), cnt = 0;
   if (n < 2) return a;
   sort(a.begin(), a.end()); // less<pair>
   vector<vec> ret;
```

```
for (int i = 0; i < n; ++i) {
 6
7
           while (cnt > 1
8
           && turn_left(ret[cnt - 2], a[i], ret[cnt - 1]))
9
            --cnt, ret.pop_back();
10
           ++cnt, ret.push_back(a[i]); }
11
       int fixed = cnt;
12
       for (int i = n - 2; i >= 0; --i) {
13
           while (cnt > fixed
           && turn_left(ret[cnt - 2], a[i], ret[cnt - 1]))
14
15
            --cnt, ret.pop_back();
16
           ++cnt, ret.push_back(a[i]); }
17
       ret.pop_back(); return ret;
   } // counter-clockwise
```

1.8 半平面交

```
struct lin {
 2
       vec s, e; db k;
 3
       lin(vec _s, vec _e): s(_s), e(_e), k(atan2((e - s).y,
         \hookrightarrow (e - s).x)) {}
4
       il vec operator()() const {return e - s;}
5
   };
6
   vec cross(const lin &l1, const lin &l2) {return l1.s + l1()
    \hookrightarrow * crs(l2.s - l1.s, l2()) / crs(l1(), l2());}
   bool cmpl(lin a, lin b) { // 极角排序, 极角相同靠左优先
       if (cmp(a.k, b.k) == 0) return sign(crs(b.e-a.s, a()))
8

→ > 0;

9
       return cmp(a.k, b.k) < 0;
10
   }
   bool Onright(lin a, lin b, lin c) { // a,b 交点在 c 右边
11
12
       vec p = cross(a, b);
13
       return sign(crs(c(), p - c.s)) <= 0;</pre>
14
   }
16
   void Halfplane(vector<lin> Ls, vector<vec> &res) { // 半平
     →面交
17
       res.clear();
18
       sort(Ls.begin(), Ls.end(), cmpl);
       deque<int> q;
19
       for (int i = 0; i < (int)Ls.size(); ++i) {</pre>
20
           if (i != 0 \&\& cmp(Ls[i].k, Ls[i-1].k) == 0)
21
              → continue;
           while (q.size() >= 2 && Onright(Ls[q[q.size() -
22
             23
           while (q.size() >= 2 && Onright(Ls[q.front()],
            24
           q.push_back(i);
25
26
       while (q.size() >= 2 && Onright(Ls[q[q.size() - 2]],
         \hookrightarrow Ls[q.back()], Ls[q.front()])) q.pop_back();
       while (q.size() \ge 2 \&\& Onright(Ls[q[0]], Ls[q[1]],
         \hookrightarrow Ls[q.back()])) q.pop_front();
28
       if (q.size() >= 2) res.push_back(cross(Ls[q.back()],
         29
       while (q.size() >= 2) {
30
           res.push_back(cross(Ls[q[0]], Ls[q[1]]));
31
           q.pop_front();
32
       }
33
```

2. Graph

2.1 图论基本知识

2.1.1 树链的交

2.1.2 带修改MST

维护少量修改的最小生成树,可以缩点缩边使暴力复杂度变低. (银川 21: 求有 16 个'某两条边中至少选一条'的限制条件的最小生成树)

找出必须边 将修改边标 $-\infty$, 在MST上的其余边为必须边, 以此缩点. 找出无用边 将修改边标 ∞ ,不在MST上的其余边为无用边,删除之. 假设修改边数为 k, 操作后图中最多剩下 k+1 个点和 2k 条边. 2.1.3 差分约束

$x_r - x_l \le c$:add(1, r, c) $x_r - x_l \ge c$:add(r, 1, -c) 2.1.4 李超线段树

添加若干条线段或直线 $(a_i,b_i)
ightarrow (a_j,b_j)$, 每次求 [l,r] 上最上面的那 条线段的值. 思想是让线段树中一个节点只对应一条直线, 如果在这个区间加入一条直线, 如果一段比原来的优, 一段比原来的劣, 那么判断一下两条线的交点, 判断哪条直线可以完全覆盖一段一半的区间, 把它保留, 另一条 直线下传到另一半区间. 时间复杂度 $O(n \log n)$.

2.1.5 Segment Tree Beats

区间 min, max, 区间求和. 以区间取 min 为例, 额外维护最大值 m, 严格次 大值 s 以及最大值个数 t. 现在假设我们要让区间 [L,R] 对 x 取 \min , 先在 线段树中定位若干个节点,对于每个节点分三种情况讨论: 1, 当 $m \le x$ 时,显然这一次修改不会对这个节点产生影响,直接退出; 2, 当 se < x < ma时,显然这一次修改只会影响到所有最大值,所以把num加上t*(x-ma), 把ma更新为x, 打上标记退出; 3, 当 $se \ge x$ 时, 无法直接更新着一个节 点的信息,对当前节点的左儿子和右儿子递归处理. 单次操作均摊复杂度 $O(\log^2 n)$.

2.1.6 二分图

最小点覆盖=最大匹配数. 独立集与覆盖集互补. 最小点覆盖构造方法: 对二分图流图求割集, 跨过的边指示最小点覆盖. Hall定理 $G=(X,Y,E), |M|=|X| \Leftrightarrow \forall S\subseteq X, |S|\leq |A(S)|.$

2.1.7 稳定婚姻问题

男士按自己喜欢程度从高到底依次向每位女士求婚,女士遇到更喜欢的男士 时就接受他,并抛弃以前的配偶.被抛弃的男士继续按照列表向剩下的女士依次求婚,直到所有人都有配偶.算法一定能得到一个匹配,而且这个匹配 一定是稳定的. 时间复杂度 $O(n^2)$.

2.1.8 三元环

对于无向边 (u,v), 如果 $\deg_u < \deg_v$, 那么连有向边 (u,v)(以点标号为 第二关键字). 枚举 x 暴力即可. 时间复杂度 $O(m\sqrt{m})$.

 $\Rightarrow F_t(i) = (F_{t-1}(i) * A + \sum_{i \to j} F_{t-1}(j) * B + \sum_{j \to i} F_{t-1}(j) * C + D *$ (i-a)) $\operatorname{mod} P$, 枚举点 a, 迭代 K 次后求得的就是 a 点所对应的 hash值, 其中 K, A, B, C, D, P 为 hash 参数, 可自选.

2.1.10 竞赛图 Landau's Theorem

n 个点竞赛图点按出度按升序排序,前 i 个点的出度之和不小于 $\frac{i(i-1)}{2}$,度 数总和等于 $\frac{n(n-1)}{2}$. 否则可以用优先队列构造出方案.

2.1.11 Ramsey Theorem R(3,3)=6, R(4,4)=18

6个人中存在3人相互认识或者相互不认识.

2.1.12 树的计数 **Prufer**序列

树和其prufer编码——对应, 一颗 n 个点的树, 其prufer编码长度为 n-2, 12 且度数为 d_i 的点在prufer 编码中出现 d_i-1 次.

由树得到序列: 总共需要 n-2 步, 第 i 步在当前的树中寻找具有最小标号 的叶子节点,将与其相连的点的标号设为Prufer序列的第i个元素 p_i ,并将此叶子节点从树中删除,直到最后得到一个长度为 n-2 的Prufer 序列和 一个只有两个节点的树.

由序列得到树: 先将所有点的度赋初值为 1, 然后加上它的编号在Prufer序 18 列中出现的次数, 得到每个点的度; 执行 n-2 步, 第 i 步选取具有最小标号 19 的度为 1 的点 u 与 v = p_i 相连, 得到树中的一条边, 并将 u 和 v 的度减一. 20最后再把剩下的两个度为1的点连边,加入到树中.

相关结论: n 个点完全图, 每个点度数依次为 $d_1,d_2,...,d_n$, 这样生成树的棵 22 树为: $\frac{(n-2)!}{(d_1-1)!(d_2-1)!...(d_n-1)!}$.

左边有 n_1 个点, 右边有 n_2 个点的完全二分图的生成树棵树为 $n_1^{n_2-1} imes 25$

m 个连通块,每个连通块有 c_i 个点,把他们全部连通的生成树方案数: 27 $(\sum c_i)^{m-2} \prod c_i$

2.1.13 有根树的计数

首先, 令 $S_{n,j} = \sum_{1 \leq j \leq n/j}$; 于是 n+1 个结点的有根树的总数为 $a_{n+1} = \frac{\sum_{j=1}^{n} j a_j S_{n-j}}{n}. \ \pm : \ a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 1$ 33 $20, a_9 = 286, a_{11} = 1842.$

2.1.14 无根树的计数

n 是奇数时,有 $a_n - \sum_i^{n/2} a_i a_{n-i}$ 种不同的无根树. n 时偶数时,有 $a_n - \sum_i^{n/2} a_i a_{n-i} + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$ 种不同的无根树.

2.1.15 生成树计数 Kirchhoff's Matrix-Tree Thoerem

39 Kirchhoff Matrix T = Deg - A, Deg 是度数对角阵, A 是邻接矩阵. 无向 40 图度数矩阵是每个点度数;有向图度数矩阵是每个点入度. 邻接矩阵 A[u][v] 表示 $u \rightarrow v$ 边个数, 重边按照边数计算, 自环不计入度

无向图生成树计数: c = |K| 的任意 $1 \land n - 1$ 阶主子式 | 有向图外向树计数: c = | 去掉根所在的那阶得到的主子式 |

2.1.16 有向图欧拉回路计数 BEST Thoerem

$$\operatorname{ec}(G) = t_w(G) \prod_{v \in V} (\deg(v) - 1)!$$

其中 \deg 为入度 (欧拉图中等于出度), $t_w(G)$ 为以 w 为根的外向树的个 数. 相关计算参考生成树计数.

欧拉连通图中任意两点外向树个数相同: $t_v(G) = t_w(G)$.

2.1.17 Tutte Matrix

Tutte matrix A of a graph G = (V, E):

$$A_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E \text{ and } i < j \\ -x_{ij} & \text{if } (i,j) \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

where x_{ij} are indeterminates. The determinant of this skewsymmetric matrix is then a polynomial (in the variables x_{ij} , i < j): this coincides with the square of the pfaffian of the matrix A and is non-zero (as a polynomial) if and only if a perfect matching exists.

2.1.18 Edmonds Matrix

Edmonds matrix A of a balanced (|U| = |V|) bipartite graph G =(U,V,E):

$$A_{ij} = \begin{cases} x_{ij} & (u_i, v_j) \in E \\ 0 & (u_i, v_j) \notin E \end{cases}$$

where the x_{ij} are indeterminates. G 有完美匹配当且仅当关于 x_{ij} 的多 项式 $det(A_{ij})$ 不恒为 0. 完美匹配的个数等于多项式中单项式的个数.

2.1.19 有向图无环定向, 色多项式

```
图的色多项式 P_G(q) 对图 G 的 q-染色计数.
Triangle K_3: x(x-1)(x-2)
Complete graph K_n: x(x-1)(x-2)\cdots(x-(n-1))
Tree with n vertices : x(x-1)^{n-1}
Cycle C_n : (x-1)^n + (-1)^n (x-1)
# acyclic orientations of an n-vertex graph G is (-1)^n P_G(-1).
```

2.2 2 SAT

4

11

35

36

38

```
int n, m, h[2000001], p, tot, t;
int ans[2000001], cnt, bel[2000001], dfn[2000001],
  \hookrightarrow low[2000001], inz[2000001], z[2000001];
char c[2000001];
struct pp {
    int to, ne;
} b[2000001];
void add(int x, int y) {
    b[++p].to = y;
    b[p].ne = h[x];
    h[x] = p;
}
void tarjan(int x) {
    dfn[x] = low[x] = ++cnt;
    z[++t] = x; inz[x] = 1;
    for (int i = h[x]; i; i = b[i].ne) {
   int v = b[i].to;
        if (!dfn[v]) {
            tarjan(v);
            low[x] = min(low[x], low[v]);
        } else if (inz[v]) {
            low[x] = min(low[x], dfn[v]);
    if (dfn[x] == low[x]) {
        tot++;
            bel[z[t]] = tot;
            inz[z[t]] = 0;
        } while (z[t--] != x);
bool solve() {
    for (int i = 1; i <= 2 * n; i++)
        if (!dfn[i]) tarjan(i);
    for (int i = 1; i <= n; i++)
        if (bel[i] == bel[i + n]) return 0;
int main() {
    scanf("%d%d", &n, &m); p = 0;
    for (int i = 1; i <= m; i++) {
```

```
int xx, yy, x, y;
scanf("%d%d%d%d", &xx, &x, &yy, &y);
42
43
44
             add(xx + (x ^ 1)*n, yy + y * n);
45
             add(yy + (y ^ 1)*n, xx + x * n);
46
47
        if (solve()) {
             printf("POSSIBLE\n");
48
             for (int i = 1; i <= n; i++) {
49
                 if (bel[i] < bel[i + n]) printf("0 ");</pre>
50
51
                 else printf("1 ");
52
        } else printf("IMPOSSIBLE\n");
53
54
55
   }
```

极大团

```
const int maxn = 129;
   int some[maxn][maxn], all[maxn][maxn], none[maxn][maxn],

    g[maxn][maxn];
   void dfs(int d, int an, int sn, int nn) {
       if (!sn && !nn) ++S;
       if (S > 1000) return; //题意表明S超过1000就输
6

→ 出Impossible

       int u = some[d][0];
8
       for (int i = 0; i < sn; ++i) {
9
           int v = some[d][i];
10
           if (g[u][v]) continue;
11
           int tsn = 0, tnn = 0;
           //for(int j=0; j<an; ++j) all[d+1][j]=all[d][j];
//all[d+1][an]=v; //可以不写这两行
12
13
           for (int j = 0; j < sn; ++j) if (g[v][some[d][j]])
14
                    some[d + 1][tsn++] = some[d][j];
15
16
           for (int j = 0; j < nn; ++j) if (g[v][none[d][j]])
17
                    none[d + 1][tnn++] = none[d][j];
18
           dfs(d + 1, an + 1, tsn, tnn);
19
           some[d][i] = 0, none[d][nn++] = v;
20
21
22
23
   int main() {
       ios::sync_with_stdio(false);
25
       while (cin >> n >> m) {
26
           memset(g, 0, sizeof(g));
27
28
           for (int i = 0; i < m; ++i) {
29
                int a, b;
                cin >> a >> b;
30
31
                g[a][b] = g[b][a] = 1;
32
33
           for (int i = 0; i < n; ++i) some[0][i] = i + 1; //
              → some的初始化
34
           dfs(0, 0, n, 0);
35
           if (S > 1000) cout << "Too many maximal sets of
             36
           else cout << S << endl;</pre>
37
       }
38
   }
```

2.4 k短路

```
#include <algorithm>
   #include <cstdio>
   #include <cstring>
   #include <queue>
   using namespace std;
   const int maxn = 200010;
   int n, m, s, t, k, x, y, ww, cnt, fa[maxn];
9
   struct Edge {
10
     int cur, h[maxn], nxt[maxn], p[maxn], w[maxn];
11
12
     void add_edge(int x, int y, int z) {
13
       cur++;
       nxt[cur] = h[x];
14
15
       h[x] = cur;
16
       p[cur] = y;
17
       w[cur] = z;
     }
18
19
   } e1, e2;
20
   int dist[maxn];
```

```
bool tf[maxn], vis[maxn], ontree[maxn];
23
   struct node {
25
     int x, v;
26
     node* operator=(node a) {
28
       x = a.x;
29
       v = a.v;
30
       return this;
31
33
     bool operator<(node a) const { return v > a.v; }
34
35
36
   priority_queue<node> Q;
37
38
   void dfs(int x) {
39
     vis[x] = true;
     for (int j = e2.h[x]; j; j = e2.nxt[j])
40
       if (!vis[e2.p[j]])
41
          if (dist[e2.p[j]] == dist[x] + e2.w[j])
42
43
            fa[e2.p[j]] = x, ontree[j] = true, dfs(e2.p[j]);
44
45
   struct LeftistTree {
46
47
     int cnt, rt[maxn], lc[maxn * 20], rc[maxn * 20],

    dist[maxn * 20];

48
     node v[maxn * 20];
49
     LeftistTree() { dist[0] = -1; }
51
52
     int newnode(node w) {
53
       cnt++;
54
       v[cnt] = w;
55
       return cnt;
56
57
58
     int merge(int x, int y) {
       if (!x \mid | !y) return x + y;
59
       if (v[x] < v[y]) swap(x, y);
61
       int p = ++cnt;
       lc[p] = lc[x];
62
       v[p] = v[x];
63
       rc[p] = merge(rc[x], y);
64
        if (dist[lc[p]] < dist[rc[p]]) swap(lc[p], rc[p]);</pre>
       dist[p] = dist[rc[p]] + 1;
66
67
       return p;
68
     }
   } st;
69
70
   void dfs2(int x) {
71
     vis[x] = true;
     if (fa[x]) st.rt[x] = st.merge(st.rt[x], st.rt[fa[x]]);
73
74
     for (int j = e2.h[x]; j; j = e2.nxt[j])
75
       if (fa[e2.p[j]] == x && !vis[e2.p[j]]) dfs2(e2.p[j]);
76
77
   int main() {
78
     scanf("%d%d%d%d%d", &n, &m, &s, &t, &k);
79
     for (int i = 1; i <= m; i++)
80
       scanf("%d%d%d", &x, &y, &ww), e1.add_edge(x, y, ww),
81
          \hookrightarrow e2.add_edge(y, x, ww);
82
     Q.push({t, 0});
     while (!Q.empty()) {
       a = Q.top();
84
85
       Q.pop();
       if (tf[a.x]) continue;
       tf[a.x] = true;
87
88
       dist[a.x] = a.v;
       for (int j = e2.h[a.x]; j; j = e2.nxt[j])
89
          \hookrightarrow Q.push({e2.p[j], a.v + e2.w[j]});
     if (k == 1) {
91
       if (tf[s])
         printf("%d\n", dist[s]);
93
94
        else
         printf("-1\n");
96
       return 0;
97
98
     dfs(t):
     for (int i = 1; i <= n; i++)
gg
100
       if (tf[i])
```

4

23

24

25

26

27

28

29

31

32

33

34

35

36

37

38

39

40

41

42

43 44

45

46

47

48

49

52

53

54

55

57

58

59

60

61

64

65

66

67

68

69

70

```
101
           for (int j = e1.h[i]; j; j = e1.nxt[j])
             if (!ontree[j])
               if (tf[e1.p[j]])
103
                 st.rt[i] = st.merge(
104
105
                     st.rt[i],
106
                     st.newnode({e1.p[j], dist[e1.p[j]] +
                       \hookrightarrow e1.w[j] - dist[i]}));
      for (int i = 1; i <= n; i++) vis[i] = false;
108
      dfs2(t):
109
      if (st.rt[s]) Q.push({st.rt[s], dist[s] +

    st.v[st.rt[s]].v});
      while (!Q.empty()) {
110
111
        a = Q.top();
112
        Q.pop();
113
        cnt++;
114
        if (cnt == k - 1) {
           printf("%d\n", a.v);
115
116
117
        if (st.lc[a.x]) // 可并堆删除直接把左右儿子加入优先队列
118
119
          Q.push({st.lc[a.x], a.v - st.v[a.x].v +}
                                                                       11

    st.v[st.lc[a.x]].v});
                                                                      12
        if (st.rc[a.x])
                                                                       13
           Q.push({st.rc[a.x], a.v - st.v[a.x].v +}
121
                                                                       14

    st.v[st.rc[a.x]].v});
                                                                       15
122
        x = st.rt[st.v[a.x].x];
                                                                      16
123
        if (x) Q.push(\{x, a.v + st.v[x].v\});
                                                                       17
124
                                                                       18
      printf("-1\n");
125
                                                                      19
126
      return 0;
                                                                       20
127
                                                                       21
```

2.5KM

```
#include<cstdio>
   #include<iostream>
   #include<cstring>
   #include<cmath>
   #include<queue>
   using namespace std;
   int n, N, M, k, d[501][501], match[501], ka[501], kb[501],

    visb[501], visa[501], p[501];

 8
   long long c[501], delta;
   void bfs(int x) {
9
10
       int a, y = 0, yy = 0;
11
       for (int i = 1; i <= n; i++)p[i] = 0, c[i] = 1e18;
12
       match[y] = x;
13
       do {
            a = match[y], delta = 1e18, visb[y] = 1;
14
            for (int b = 1; b <= n; b++) {
15
16
                if (!visb[b]) {
17
                     if (c[b] > ka[a] + kb[b] - d[a][b])
                         c[b] = ka[a] + kb[b] - d[a][b], p[b] =
18
                     if (c[b] < delta)</pre>
19
20
                         delta = c[b], yy = b;
21
22
            for (int b = 0; b <= n; b++) {
23
                if (visb[b]) {
24
25
                     ka[match[b]] -= delta, kb[b] += delta;
                } else c[b] -= delta;
26
27
            }
28
            y = yy;
       } while (match[y]);
29
30
       while (y)match[y] = match[p[y]], y = p[y];
31
32
   long long KM() {
33
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) visb[j] = 0;
34
35
            bfs(i);
36
37
       long long ans = 0;
       for (int i = 1; i <= n; i++) ans += d[match[i]][i];</pre>
38
39
       return ans;
40
   }
   int main() {
41
       scanf("%d%d%d", &N, &M, &k);
42
43
       n = max(N, M);
       while (k--) {
44
            int x, y, z;
scanf("%d%d%d", &x, &y, &z);
45
46
```

```
d[y][x] = z;
48
         printf("%lld\n", KM());
49
         for (int i = 1; i <= N; i++)
printf("%d ", (d[match[i]][i] == 0) ? 0 :
50
51

    match[i]);

         return 0;
52
53
```

```
2.6
       tarjan
   // 强连通分量
   void tarjan(int x) {
       dfn[x] = low[x] = ++tot;
       s[++len] = x;
       instack[x] = 1;
       for (int i = head[x]; i; i = e[i].next) {
           int y = e[i].to;
           if (!dfn[y]) {
               tarjan(v);
               low[x] = min(low[x], low[y]);
           } else {
               if (instack[y])low[x] = min(low[x], low[y]);
       if (dfn[x] == low[x]) {
           cnt++
           ans[cnt].push_back(x);
           while (s[len] != x) {
               belong[s[len]] = cnt;
               instack[s[len]] = 0;
               ans[cnt].push_back(s[len]);
               len--;
           len--;
           instack[x] = 0;
           belong[x] = cnt;
   // 边双
   void tarjan(int x, int las) {
       low[x] = dfn[x] = ++cnt;
       st.push(x);
       for (auto i : e[x]) {
           if (i == las) continue;
           if (!dfn[i]) {
               tarjan(i, x);
               low[x] = min(low[x], low[i]);
           } else low[x] = min(low[x], dfn[i]);
       if (dfn[x] == low[x]) {
           vector<int> vec:
           vec.push back(x);
           while (st.top() != x) {
               vec.push_back(st.top());
               st.pop();
           st.pop();
           ans.push_back(vec);
50
   // 点双
51
   void tarjan(int x, int root) { //求割点的改版(其实不需
     → 要root)
       dfn[x] = low[x] = ++cnt;
       if (x == root && !head[x]) { //孤立点判定
           dcc[++ans].push_back(x);
       sta.push(x);
       for (int i = head[x]; i; i = nxt[i]) {
           int g = go[i];
           if (!dfn[g]) {
               tarjan(g, root);
low[x] = min(low[x], low[g]);
               if (low[g] >= dfn[x]) {
                   ans++:
                   int p;
                   do { //弹栈
                       p = sta.top();
                       sta.pop();
                       dcc[ans].push_back(p);
                   } while (p != g); //注意此处, 因为要求是不到
                     → 达出点
```

```
71
                   dcc[ans].push_back(x);//别忘了加入源点!
72
               }
73
           } else
               low[x] = min(low[x], dfn[g]);
74
75
76
   }
```

最小斯坦纳树

```
//给定一个带边权的无向连通图G, 再给定包含k个结点的点集S, 选
     →出G的子图G', 使得G'包含S, G'为连通图, 且G'边权和最小
   #include<bits/stdc++.h>
   #define mp make_pair
   #define zjx printf("%d",
   #define AK dp[c[1]][(1 << k)-1]
   #define IOI );
   using namespace std;
   int n, m, k, p, h[101], dp[101][1024], c[11], vis[101];
   struct tree {
10
       int to, ne, v;
11
   } a[1001];
12
   void add(int x, int y, int z) {
13
       a[++p].to = y;
       a[p].ne = h[x];
14
15
       a[p].v = z;
16
       h[x] = p;
17
18
   priority_queue<pair<int, int>, vector<pair<int, int> >,

    greater<pair<int, int> > >q;
19
   void dijkstra(int s) {
20
       memset(vis, 0, sizeof(vis));
21
       while (!q.empty()) {
           int x = q.top().second;
22
23
           q.pop();
24
           if (vis[x])continue;
25
           vis[x] = 1;
26
           for (int i = h[x]; i; i = a[i].ne) {
27
               if (dp[a[i].to][s] > dp[x][s] + a[i].v) {
28
                    dp[a[i].to][s] = dp[x][s] + a[i].v;
29
                    q.push(mp(dp[a[i].to][s], a[i].to));
30
               }
31
           }
32
       }
33
   int main() {
34
       scanf("%d%d%d", &n, &m, &k);
35
36
       for (int i = 1; i <= m; i++) {
           int x, y, z;
scanf("%d%d%d", &x, &y, &z);
37
38
39
           add(x, y, z), add(y, x, z);
40
       memset(dp, 0x3f, sizeof(dp));
41
       for (int i = 1; i <= k; i++)scanf("%d", &c[i]),
42
         \hookrightarrow dp[c[i]][1 \leftrightarrow (i - 1)] = 0;
43
       for (int s = 1; s < (1 << k); s++) {
44
           for (int i = 1; i <= n; i++) {
45
               for (int ss = s & (s - 1); ss; ss = (ss - 1)&s)
46
                    dp[i][s] = min(dp[i][s], dp[i][ss] + dp[i]
                      if (dp[i][s] != 0x3f3f3f3f)q.push(mp(dp[i][s],
47
                 → i));
48
49
           dijkstra(s);
50
       zjx AK IOI
51
52
       return 0;
53
   }
```

3. Data Structure

3.1 LCT 动态树

```
10
   namespace LCT {
 1
       int ch[N][2], f[N], sum[N], val[N], tag[N], dat[N]; //
                                                                 11
 2
         → dat 维护的链信息, val 点上信息
                                                                 12
       inline void PushUp(int x) {
                                                                 13
           dat[x] = dat[ch[x][0]] ^ dat[ch[x][1]] ^ val[x];
                                                                  14
 4
                                                                 15
 5
                                                                 16
 6
       inline void PushRev(int x) {swap(ch[x][0], ch[x][1]);
         17
       inline void PushDown(int x) {
           if (tag[x] == 0) return;
8
                                                                 18
9
           PushRev(ch[x][0]); PushRev(ch[x][1]); tag[x] = 0;
10
                                                                  19
                                                                 20
       inline bool Get(int x) {return ch[f[x]][1] == x;} // 是
11
         → 父亲的哪个儿子
                                                                  21
12
       inline bool IsRoot(int x) {return (ch[f[x]][1] != x \&\&
         → ch[f[x]][0] != x);} // 是否是当前 Splay 的根
       inline void Rotate(int x) { // Splay 旋转
                                                                  22
13
                                                                  23
14
           int y = f[x], z = f[y], k = Get(x);
15
           if (!IsRoot(y)) ch[z][Get(y)] = x;
                                                                 24
                                                                 25
           ch[y][k] = ch[x][k ^ 1]; f[ch[x][k ^ 1]] = y;
16
           ch[x][k ^ 1] = y; f[y] = x; f[x] = z;
                                                                 26
17
           PushUp(y); PushUp(x);
                                                                  27
18
                                                                 28
19
20
       void Updata(int x) { // Splay 中从上到下 PushDown
           if (!IsRoot(x)) Updata(f[x]);
                                                                  29
21
22
           PushDown(x);
23
                                                                  30
                                                                  31
24
       inline void Splay(int x) { // Splay 上把 x 转到根
                                                                  32
25
           Updata(x);
                                                                 33
           for (int fa; fa = f[x], !IsRoot(x); Rotate(x)) {
26
               if (!IsRoot(fa)) Rotate(Get(fa) == Get(x) ? fa
                                                                 34
                                                                  35
                 \hookrightarrow : x);
                                                                 36
28
                                                                 37
29
           PushUp(x);
                                                                  38
30
       inline void Access(int x) { // 辅助树上打通 x 到根的路径
31
         → (即 x 到根变为实链)
                                                                  39
32
           for (int p = 0; x; p = x, x = f[x]) {
               Splay(x); ch[x][1] = p; PushUp(x);
                                                                  40
33
                                                                  41
34
                                                                  42
35
       inline void MakeRoot(int x) { // 钦定 x 为辅助树根
                                                                  43
36
37
           Access(x); Splay(x); PushRev(x);
38
                                                                  44
39
       inline int FindRoot(int x) { // 找 x 所在辅助树根
40
           Access(x); Splay(x);
                                                                  45
           while (ch[x][0]) PushDown(x), x = ch[x][0];
41
42
           Splay(x); // 不加复杂度会假
                                                                  46
43
                                                                  47
           return x;
                                                                  48
44
                                                                 49
45
       inline void Split(int x, int y) { // 把 x 到 y 的路径提
         → 出来, 并以 y 为 Splay 根
                                                                  50
46
           MakeRoot(x); Access(y); Splay(y);
                                                                  51
47
                                                                 52
48
       inline bool Link(int x, int y) { // 连接 x,y 两点
49
           MakeRoot(x);
                                                                  53
50
           if (FindRoot(y) == x) return false;
51
           f[x] = y;
52
           return true;
53
       inline bool Cut(int x, int y) { // x,y 断边
54
           MakeRoot(x);
55
                                                                  56
56
           if (FindRoot(y) == x && f[y] == x && !ch[y][0]) {
57
               f[y] = ch[x][1] = 0; PushUp(x);
                                                                 57
58
               return true;
                                                                 58
59
           }
                                                                 59
60
           return false;
61
       }
                                                                 60
                                                                 61
62
   }
                                                                 62
```

3.2 KD Tree

```
// KDTree 二维平面邻域查询 K 远点对 n=1e5 k=100
  priority_queue<ll, vector<ll>, greater<ll> >q; // 小根堆
3
  namespace KDTree {
      struct node {
          int X[2];
          int &operator[](const int k) {return X[k];}
6
      } p[N];
8
      int nowd:
```

```
bool cmp(node a, node b) {return a.X[nowd] <</pre>
      \hookrightarrow b.X[nowd];}
    int lc[N], rc[N], L[N][2], R[N][2]; // lc/rc 左右孩子;
      → L/R 对应超矩形各个维度范围
    inline ll sqr(int x) {return 111 * x * x;}
    void pushup(int x) { // 更新该点所代表空间范围
        L[x][0] = R[x][0] = p[x][0];
        L[x][1] = R[x][1] = p[x][1];
        if (lc[x]) {
             umin(L[x][0], L[lc[x]][0]); umax(R[x][0],
               \hookrightarrow R[lc[x]][0]);
             umin(L[x][1], L[lc[x]][1]); umax(R[x][1],
               \hookrightarrow R[lc[x]][1]);
        if (rc[x]) {
             umin(L[x][0], L[rc[x]][0]); umax(R[x][0],
               \hookrightarrow R[rc[x]][0]);
             umin(L[x][1], L[rc[x]][1]); umax(R[x][1],
              \hookrightarrow R[rc[x]][1]);
    int build(int 1, int r) {
        if (1 > r) return 0;
        int mid = (1 + r) >> 1;
         // >>> 方差建树
        db av[2] = {0, 0}, va[2] = {0, 0}; // av 平均数, va
          → 方差
         for (int i = 1; i \leftarrow r; ++i) av[0] += p[i][0],
          \hookrightarrow av[1] += p[i][1];
        av[0] /= (r - l + 1); av[1] /= (r - l + 1);
        for (int i = 1; i <= r; ++i) {
             va[0] += sqr(av[0] - p[i][0]);
             va[1] += sqr(av[1] - p[i][1]);
        if (va[0] > va[1]) nowd = 0;
        else nowd = 1; // 找方差大的维度划分
         // >>> 轮换建树 nowd=dep%D
        nth_element(p + l, p + mid, p + r + 1, cmp); // 以
          → 该维度中位数分割
        lc[mid] = build(1, mid - 1); rc[mid] = build(mid +
          \hookrightarrow 1, r);
        pushup(mid);
        return mid;
    ll dist(int a, int x) { // 估价函数, 点 a 到树上 x 点对应
      →空间最远距离
        return max(sqr(p[a][0] - L[x][0]), sqr(p[a][0] -
          \hookrightarrow R[x][0])) +
                \max(\text{sqr}(p[a][1] - L[x][1]), \text{sqr}(p[a][1] -
                  \hookrightarrow R[x][1]);
    void query(int l, int r, int a) { // 点 a 邻域查询
        if (1 > r) return;
        int mid = (1 + r) >> 1;
        11 t = sqr(p[mid][0] - p[a][0]) + sqr(p[mid][1] -
          \hookrightarrow p[a][1]);
        if (t > q.top()) q.pop(), q.push(t); // 更新答案
        11 disl = dist(a, lc[mid]), disr = dist(a,

¬ rc[mid]);

        if (disl > q.top() && disr > q.top()) // 两边都有机
          → 会更新,优先搜大的
             (disl > disr)? (query(l, mid - 1, a),
               \hookrightarrow query(mid + 1, r, a)) : (query(mid + 1, r,
              \hookrightarrow a), query(1, mid - 1, a));
             (disl > q.top()) ? query(1, mid - 1, a) :
               \hookrightarrow query(mid + 1, r, a);
    }
using namespace KDTree;
int main() {
    red(n); red(k); k *= 2;
    for (int i = 1; i <= k; ++i) q.push(0);
    for (int i = 1; i <= n; ++i) red(p[i][0]), red(p[i]</pre>

    [1]);
    build(1, n);
    for (int i = 1; i <= n; ++i) query(1, n, i);
    printf("%lld\n", q.top());
```

```
// 动态 KDTree 维护空间权值 (单点修改 & 空间查询)
2 // 时间复杂度 O(log n) ~ O(n^(1-1/k))
```

64

65

66

67

```
#define sqr(x) ((x) * (x))
   namespace KDT {
4
 5
       struct dat {
6
            int X[2];
7
            int &operator[](const int k) {return X[k];}
8
       db alp = 0.725; // 重构常数
9
10
       int nowd;
       bool cmp(int a, int b) {return p[a][nowd] < p[b]</pre>
11
          // root: 根 cur: 总点数 d: 当前分割维度 lc/rc: 左右儿子
12
         → L/R: 当前空间范围 siz: 子树大小 sum/val 空间的值,单
         → 点的值
       int root, cur, d[N], lc[N], rc[N], L[N][2], R[N][2],
13
          \hookrightarrow siz[N], sum[N], val[N];
       int g[N], t; // 用于重构的临时数组
14
       void pushup(int x) {
16
            siz[x] = siz[lc[x]] + siz[rc[x]] + 1;
17
            sum[x] = sum[lc[x]] + sum[rc[x]] + val[x];
18
            L[x][0] = R[x][0] = p[x][0];
19
            L[x][1] = R[x][1] = p[x][1];
20
            if (lc[x]) {
21
                umin(L[x][0], L[lc[x]][0]); umax(R[x][0],
                  \hookrightarrow R[lc[x]][0]);
                umin(L[x][1], L[lc[x]][1]); umax(R[x][1],
                  \hookrightarrow R[lc[x]][1]);
23
24
            if (rc[x]) {
                umin(L[x][0], L[rc[x]][0]); umax(R[x][0],
25
                  \hookrightarrow R[rc[x]][0]);
26
                umin(L[x][1], L[rc[x]][1]); umax(R[x][1],
                  \hookrightarrow R[rc[x]][1]);
27
            }
28
29
       int build(int l, int r) { // 对 g[1...t] 进行建树 , 对应
         → 点都是 g[x]。方差建树
30
            if (1 > r) return 0;
31
            int mid = (1 + r) >> 1;
32
            db av[2] = \{0, 0\}, va[2] = \{0, 0\};
            for (int i = 1; i <= r; ++i) av[0] += p[g[i]][0],
33
              \hookrightarrow av[1] += p[g[i]][1];
34
            av[0] /= (r - l + 1); av[1] /= (r - l + 1);
            for (int i = 1; i \le r; ++i) va[0] += sqr(av[0] -
35
              \hookrightarrow p[g[i]][0]), va[1] += sqr(av[1] - p[g[i]][1]);
36
            if (va[0] > va[1]) d[g[mid]] = nowd = 0;
37
            else d[g[mid]] = nowd = 1;
38
            nth_element(g + l, g + mid, g + r + 1, cmp);
39
            lc[g[mid]] = build(1, mid - 1); rc[g[mid]] =
              \hookrightarrow build(mid + 1, r);
40
            pushup(g[mid]);
            return g[mid];
41
42
43
       void expand(int x) { // 将子树展开到临时数组里
44
            if (!x) return;
45
            expand(lc[x]);
46
            g[++t] = x;
47
            expand(rc[x]);
48
49
       void rebuild(int &x) { // x 所在子树重构
50
            t = 0; expand(x);
            x = build(1, t);
51
52
       bool chk(int x) {return alp * siz[x] <=</pre>
53
         → (db)max(siz[lc[x]], siz[rc[x]]);} // 判断失衡
       void insert(int &x, int a) { // 插入点 a , p[a],val[a]
54
         → 为其信息
            if (!x) \{ x = a; pushup(x); d[x] = rand() \& 1;

    return; }

56
            if (p[a][d[x]] \leftarrow p[x][d[x]]) insert(lc[x], a);
57
            else insert(rc[x], a);
58
            pushup(x);
59
            if (chk(x)) rebuild(x); // 失衡暴力重构
60
       dat Lt, Rt; // 询问一块空间的值 (为了减小常数把参数放在外
61
         ⇔面)
62
       int query(int x) {
63
            if (!x || Rt[0] < L[x][0] || Lt[0] > R[x][0] ||
              \hookrightarrow Rt[1] < L[x][1]
64
                || Lt[1] > R[x][1]) return 0; // 结点为空或与询
                  → 问取间无交
65
            if (Lt[0] \leftarrow L[x][0] \&\& R[x][0] \leftarrow Rt[0] \&\& Lt[1]
              \hookrightarrow \leftarrow L[x][1]
```

```
66
                 && R[x][1] <= Rt[1]) return sum[x]; // 区间完全
             int ret = 0;
68
             if (Lt[0] \leftarrow p[x][0] \&\& p[x][0] \leftarrow Rt[0] \&\& Lt[1]
               \hookrightarrow \langle = p[x][1]
69
                 && p[x][1] <= Rt[1]) ret += val[x]; // 当前点在
                    ⊶ 区间内
70
             return query(lc[x]) + query(rc[x]) + ret;
71
72
   }
73
   using namespace KDT;
74
   int main() {
        int n; read(n);
75
        for (int op;;) {
76
77
             read(op);
78
             switch (op) {
79
             case 1:
80
                 ++cur; read(p[cur][0]); read(p[cur][1]);

    read(val[cur]);
81
                 insert(root, cur);
82
                 break;
83
             case 2:
                 read(Lt[0]); read(Lt[1]); read(Rt[0]);
                   \hookrightarrow read(Rt[1]);
                 printf("%d\n", query(root));
86
                 break:
87
             case 3: | return 0; break;
88
89
        return 0;
90
91
   }
```

3.3 李超线段树

```
// 李超线段树 对于 (x1,y1) (x2,y2) -> y=0*x+max(y1,y2)
     \hookrightarrow [x1,x1]
   #define ls (x<<1)
   #define rs (x<<1|1)
   typedef long long 11;
   typedef double db;
   const int N = 100010;
   const int M = 40000;
   struct line {
       db k, b;
9
10
   } lin[N];
   db val(int id, db X) {return lin[id].k * X + lin[id].b;}
11
12
   int D[N << 2], n, id;</pre>
   void modify(int L, int R, int id, int l = 1, int r = M - 1,
     → int x = 1) { // 线 lin[id], 范围 [L, R]
       if (L <= 1 && r <= R) \{
14
            int mid = (1 + r) >> 1, lid = D[x];
15
            db lst = val(D[x], mid), now = val(id, mid);
16
17
            if (1 == r) \{ if (now > lst) D[x] = id; return ; \}
            if (lin[id].k > lin[D[x]].k) {
18
19
                if (now > lst) D[x] = id, modify(L, R, lid, l)
                  \hookrightarrow mid, ls); // id->lid
20
                else modify(L, R, id, mid + 1, r, rs);
            } else if (lin[id].k < lin[D[x]].k) {</pre>
21
                if (now > lst) D[x] = id, modify(L, R, lid, mid)
                  ← + 1, r, rs); // id->lid
                else modify(L, R, id, l, mid, ls);
23
            } else if (lin[id].b > lin[D[x]].k) D[x] = id;
24
            return ;
25
26
        int mid = (l + r) \gg 1;
27
       if (L <= mid) modify(L, R, id, l, mid, x << 1);
28
29
       if (R > mid) modify(L, R, id, mid + 1, r, x \leftrightarrow 1 | 1);
30
31
   int gmax(int x, int y, int ps) {
32
       if (val(x, ps) > val(y, ps)) return x;
       if (val(x, ps) < val(y, ps)) return y;</pre>
33
34
       return (x < y) ? x : y;
35
   }
36
   int query(int ps, int l = 1, int r = M - 1, int x = 1) { //
     → 查 x=ps
37
       if (l == r) return D[x];
38
        int mid = (1 + r) >> 1, ret = D[x], t = 0;
39
       if (ps <= mid)</pre>
40
            t = query(ps, 1, mid, 1s);
41
42
43
            t = query(ps, mid + 1, r, rs);
44
        return gmax(ret, t, ps);
```

```
45 }
  3.4
        吉司机线段树
 1
    * seg-beats 吉司机线段树
    * 区间最值操作
3
      支持 区间取min, 区间取max, 区间加减, 区间求和, 区间最小/大
    * 复杂度 O(m log n)
6
   #define ls (x << 1)
8
   #define rs (x << 1 | 1)
   #define mid ((l + r) >> 1)
   typedef long long 11;
   const int N = 500010;
11
12
   const int inf = 0x3f3f3f3f;
13
   struct datmn {
       int fi, se, cnt; // 最小值, 次小值, 最小值个数
14
15
       datmn() {fi = se = inf; cnt = 0;}
       void ins(int x, int c) {
16
17
           if (x < fi) se = fi, cnt = c, fi = x;
18
           else if (x == fi) cnt += c;
           else if (x < se) se = x;
19
20
21
       friend datmn operator+(const datmn &a, const datmn &b)
22
           datmn r = a; r.ins(b.fi, b.cnt); r.ins(b.se, 0);

    return r;

23
24
   };
25
   struct datmx {
26
       int fi, se, cnt;
27
       datmx() {fi = se = -inf; cnt = 0;}
28
       void ins(int x, int c) {
           if (x > fi) se = fi, cnt = c, fi = x;
29
30
           else if (x == fi) cnt += c;
           else if (x > se) se = x;
31
32
33
       friend datmx operator+(const datmx &a, const datmx &b)
           datmx r = a; r.ins(b.fi, b.cnt); r.ins(b.se, 0);
             \hookrightarrow return r;
35
36
   };
37
38
   struct node {
39
       datmn mn; datmx mx;
       11 sum; int addmn, addmx, add, len;
40
41
   } t[N << 2];
42
   int n, m, a[N];
43
   void pushup(int x) {
       t[x].mx = t[ls].mx + t[rs].mx;
44
45
       t[x].mn = t[ls].mn + t[rs].mn;
46
       t[x].sum = t[ls].sum + t[rs].sum;
47
   }
48
   void build(int l = 1, int r = n, int x = 1) {
49
       t[x].add = t[x].addmn = t[x].addmx = 0;
       t[x].len = r - l + 1;
50
51
       if (1 == r) {
52
           t[x].mx = datmx(); t[x].mx.ins(a[1], 1);
53
           t[x].mn = datmn(); t[x].mn.ins(a[1], 1);
54
           t[x].sum = a[1];
55
56
57
       build(1, mid, ls); build(mid + 1, r, rs);
58
       pushup(x);
59
   void update(int x, int vn, int vx, int v) { // vn: addmn,
     \hookrightarrow vx: addmx, v: add
       // 所有数相同特判, 此时最大值 tag 和最小值 tag 应该相同且
61
         → 不等于其他值 tag
62
       if (t[x].mn.fi == t[x].mx.fi) {
63
           if (vn == v) vn = vx;
           else vx = vn;
64
           t[x].sum += (11)vn * t[x].mn.cnt;
65
66
       } else t[x].sum += (ll)vn * t[x].mn.cnt + (ll) vx *
         \hookrightarrow t[x].mx.cnt + (ll)v * (t[x].len - t[x].mn.cnt -
         \hookrightarrow t[x].mx.cnt);
       if (t[x].mn.se == t[x].mx.fi) t[x].mn.se += vx; // 次小
67
         → 值 = 最大值,应该用最大值 tag 处理
       else if (t[x].mn.se != inf) t[x].mn.se += v;
68
69
       if (t[x].mx.se == t[x].mn.fi) t[x].mx.se += vn; // 次大
         → 信同理
```

```
70
        else if (t[x].mx.se != -inf) t[x].mx.se += v;
71
        t[x].mn.fi += vn; t[x].mx.fi += vx;
72
        t[x].addmn += vn; t[x].addmx += vx; t[x].add += v;
73
74
    void pushdown(int x) {
75
        int mn = min(t[ls].mn.fi, t[rs].mn.fi);
        int mx = max(t[ls].mx.fi, t[rs].mx.fi);
76
        update(ls, (mn == t[ls].mn.fi) ? t[x].addmn : t[x].add,
77
          \hookrightarrow (mx == t[ls].mx.fi) ? t[x].addmx : t[x].add,
           \hookrightarrow t[x].add);
78
        update(rs, (mn == t[rs].mn.fi) ? t[x].addmn : t[x].add,
          \hookrightarrow (mx == t[rs].mx.fi) ? t[x].addmx : t[x].add,
          \hookrightarrow t[x].add);
        t[x].add = t[x].addmn = t[x].addmx = 0;
79
80
    void modifyadd(int L, int R, int v, int l = 1, int r = n,
81
      \hookrightarrow int x = 1) {
        if (r < L \mid\mid R < 1) return;
        if (L <= 1 && r <= R) return update(x, v, v, v);
83
85
        modifyadd(L, R, v, 1, mid, ls);
86
        modifyadd(L, R, v, mid + 1, r, rs);
        pushup(x);
88
    void modifymin(int L, int R, int v, int l = 1, int r = n,
89
      \hookrightarrow int x = 1) {
90
        if (r < L || R < 1) return;
91
        if (L \le 1 \&\& r \le R \&\& v > t[x].mx.se) {
             if (v >= t[x].mx.fi) return;
92
             update(x, 0, v - t[x].mx.fi, 0);
93
94
95
        pushdown(x);
96
        modifymin(L, R, v, l, mid, ls);
97
98
        modifymin(L, R, v, mid + 1, r, rs);
99
        pushup(x);
100
101
    void modifymax(int L, int R, int v, int l = 1, int r = n,
      \hookrightarrow int x = 1) {
        if (r < L \mid\mid R < 1) return;
        if (L \le 1 \&\& r \le R \&\& v < t[x].mn.se) {
103
04
             if (v <= t[x].mn.fi) return;</pre>
             update(x, v - t[x].mn.fi, 0, 0);
105
106
             return;
107
108
        pushdown(x);
109
        modifymax(L, R, v, 1, mid, ls);
110
        modifymax(L, R, v, mid + 1, r, rs);
        pushup(x);
112
    }
113
    int querymax(int L, int R, int l = 1, int r = n, int x = 1)
        if (r < L || R < 1) return -inf;</pre>
114
        if (L <= 1 && r <= R) return t[x].mx.fi;
116
        pushdown(x);
117
        return max(querymax(L, R, 1, mid, 1s), querymax(L, R,
          \hookrightarrow mid + 1, r, rs));
118
119
    int querymin(int L, int R, int l = 1, int r = n, int x = 1)
        if (r < L \mid \mid R < 1) return inf;
120
        if (L <= 1 && r <= R) return t[x].mn.fi;</pre>
21
122
        pushdown(x);
123
        return min(querymin(L, R, 1, mid, 1s), querymin(L, R,
          \hookrightarrow mid + 1, r, rs));
124
125 | ll querysum(int L, int R, int l = 1, int r = n, int x = 1)
26
        if (r < L || R < 1) return 0;
        if (L <= 1 && r <= R) return t[x].sum;</pre>
127
28
        pushdown(x);
129
        return querysum(L, R, 1, mid, ls) + querysum(L, R, mid
          \hookrightarrow + 1, r, rs);
130 }
```

3.5 FHQ Treep

```
7 // fhq - treap 简易模板
2 #define ls(p) t[p].l
3 #define mid ((l+r)>>1)
4 #define mid ((l+r)>>1)
5 using namespace std;
```

```
const int N = 100010;
   mt19937 rd(random_device{}());
   struct node {
9
       int 1, r, siz, rnd, val, tag;
10
   } t[N]; int tot, root;
11
   /* 节点回收
12
   int cyc[N],cyccnt;
   inline void delnode(int p) {cyc[++cyccnt]=p;}
13
   inline void newnode(int val) {
14
15
       int id=(cyccnt>0)?cyc[cyccnt--]:++tot;
16
       t[id]={0,0,1,(int)(rd()),val}; return id;
17
18
   inline int newnode(int val) { t[++tot] = \{0, 0, 1, (int)\}
19
     20
   inline void updata(int p) {
       t[p].siz = t[ls(p)].siz + t[rs(p)].siz + 1;
21
22
       /* maintain */
23
24
   inline void pushtag(int p, int v1) { /* tag to push */ }
25
   inline void pushdown(int p) {
26
       if (t[p].tag != std_tag) {
27
           if (ls(p)) pushtag(ls(p), t[p].tag);
28
           if (rs(p)) pushtag(rs(p), t[p].tag);
29
           t[p].tag = std_tag;
30
31
   }
32
   int merge(int p, int q) {
       if (!p || !q) return p + q;
33
       if (t[p].rnd < t[q].rnd) {
34
35
           pushdown(p);
36
           rs(p) = merge(rs(p), q);
           updata(p); return p;
37
       } else {
38
39
           pushdown(q);
40
           ls(q) = merge(p, ls(q));
41
           updata(q); return q;
42
       }
43
   }
   void split(int p, int k, int &x, int &y) {
45
       if (!p) x = 0, y = 0;
46
       else {
           pushdown(p);
47
48
           if (t[ls(p)].siz >= k) y = p, split(ls(p), k, x,
49
           else x = p, split(rs(p), k - t[ls(p)].siz - 1,
             \hookrightarrow rs(p), y);
50
           updata(p);
51
52
   }
53
   int build(int 1, int r) { // build tree on a[l..r], return
       if (1 > r) return 0;
54
55
       return merge(build(1, mid - 1), merge(newnode(a[mid]),
         \hookrightarrow build(mid + 1, r)));
56
   }
```

3.6 哈希表

```
typedef long long 11;
   const int M = 19260817;
   const int MAX_SIZE = 2000000;
   struct Hash_map {
       struct data {
6
           int nxt;
7
           11 key, value; // (key,value)
8
       } e[MAX_SIZE];
9
       int head[M], size;
10
       inline int f(ll key) { return key % M; }
11
       11 &operator[](const 11 &key) {
12
           int ky = f(key);
13
           for (int i = head[ky]; i != -1; i = e[i].nxt)
               if (e[i].key == key) return e[i].value;
14
           return e[++size] = data{head[ky], key, 0}, head[ky]
             16
17
       void clear() {
18
           memset(head, -1, sizeof(head));
19
           size = 0;
20
21
       Hash_map() {clear();}
  };
22
```

4. String

4.1 最小表示法

```
//n为串长, a下标从0开始
   int Min_show(int *a, int n) {
       int i = 0, j = 1, k = 0;
       while (i < n && j < n && k < n) \{
           auto u = a[(i + k) \% n];
           auto v = a[(j + k) \% n];
           if (u == v) ++k;
8
           else {
9
                if (u > v) i += k + 1;
               else j += k + 1;
10
                if (i == j) ++j;
11
12
               k = 0;
13
14
15
       return min(i, j);
16
```

4.2 AC 自动机

```
int son[M][26], fail[M], cnt = 0;
   void ins(const char *s) {
       int p = 0, n = strlen(s + 1);
3
       for (int i = 1; i <= n; ++i) {
           int c = s[i] - 'a';
           if (!son[p][c]) son[p][c] = ++cnt;
           p = son[p][c];
8
9
   }
   queue<int> q;
10
11
   void get_fail() {
12
       for (int c = 0; c < 26; ++c)
13
           if (son[0][c]) q.push(son[0][c]);
14
       while (!q.empty())
           int x = q.front(); q.pop();
15
16
           for (int c = 0; c < 26; ++c) {
17
                if (son[x][c]) {
                    fail[son[x][c]] = son[fail[x]][c];
18
19
                    q.push(son[x][c]);
20
                } else son[x][c] = son[fail[x]][c];
21
22
       }
23
   }
24
```

4.3 回文树

```
int len[M], fa[M], son[M][26], lst, cnt, f[M];
   char s[M];
   int extend(int n) {
       int p = lst, c = s[n] - 'a';
4
       while (s[n - len[p] - 1] != s[n]) p = fa[p];
 5
 6
       if (!son[p][c]) {
           int now = p;
           len[++cnt] = len[p] + 2;//回文串长度
 8
           p = fa[p];
           while (s[n - len[p] - 1] != s[n]) p = fa[p];
10
           fa[cnt] = son[p][c];
11
12
           lst = son[now][c] = cnt;
           f[cnt] = f[fa[cnt]] + 1;//回文串数量
13
       } else lst = son[p][c];
14
       return f[lst];
15
16
   }
   int main() {
17
       fa[0] = cnt = 1;
18
19
       val[1] = -1;
20
```

4.4 Manacher

```
char s[M << 1];
int p[M];
///为串长, a下标从1开始, p为回文串半径 (0~2n+1)
void Manacher(const char *a, int n) {
    int r = 0, mid;
    for (int i = 1; i <= n; ++i) s[i << 1] = a[i];
    for (int i = 0; i <= n; ++i) s[i * 2 + 1] = '#';
    s[0] = '#'; n = n << 1 | 1;
    for (int i = 1; i <= n; ++i) {
```

```
p[i] = (i \le r ? min(p[mid * 2 - i], p[mid] + mid -
10
              while (s[i - p[i] - 1] == s[i + p[i] + 1]) ++p[i];
11
12
            if (i + p[i] > r) r = i + p[i], mid = i;
                                                                    12
13
                                                                    13
14
   }
                                                                    14
  4.5 字符串哈希
   const int HA = 2;
                                                                    17
   const int PP[] = {318255569, 66604919, 19260817}, QQ[] =
     18
   int pw[HA][N];
                                                                    20
   void HashInit() {
                                                                    21
       for (int h = 0; h < HA; h++) {
            pw[h][0] = 1;
 8
            for (int i = 1; i < N; i++)
                pw[h][i] = (LL)pw[h][i - 1] * PP[h] % QQ[h];
9
                                                                    24
10
                                                                    25
11
                                                                    26
   }
12
   struct Hash {
                                                                    27
13
       int hs[HA], len;
       Hash() {
14
                                                                    29
15
            memset(hs, 0, sizeof hs);
16
           len = 0;
                                                                    31
17
                                                                    32
18
       Hash(int x) {
                                                                    33
            for (int h = 0; h < HA; h++) hs[h] = x;
19
                                                                    34
20
21
       Hash operator + (const int &x)const {
           Hash res;
24
            res.len = len + 1;
                                                                    39
25
            for (int h = 0; h < HA; h++)
                res.hs[h] = ((LL)hs[h] * PP[h] + x) % QQ[h];
26
                                                                    41
27
            return res;
                                                                    42
28
                                                                    43
29
       Hash operator - (const Hash &x)const {
                                                                    44
                                                                    45
                                                                       }
30
            Hash res;
            res.len = len - x.len;
31
32
            for (int h = 0; h < HA; h++) {
                res.hs[h] = (hs[h] - (LL)pw[h][res.len] *
33
                  \hookrightarrow x.hs[h]) \% QQ[h];
                if (res.hs[h] < 0) res.hs[h] += QQ[h];
35
36
            return res;
37
38
       bool operator == (const Hash &x)const {
            for (int h = 0; h < HA; h++)
39
                if (hs[h] != x.hs[h]) return false;
40
41
            return len == x.len;
42
       \ensuremath{//} below : not that frequently used
43
       Hash operator + (const Hash &x)const {
45
           Hash res;
46
            res.len = len + x.len;
47
            for (int h = 0; h < HA; h++)
                res.hs[h] = ((LL)hs[h] * pw[h][x.len] +
48
                                                                    16
                 \hookrightarrow x.hs[h]) \% QQ[h];
                                                                    17
49
            return res;
                                                                    18
50
                                                                    19
51
   } H:
52
   Hash operator + (const int &a, const Hash &b) {
                                                                    21
53
       Hash res;
54
       res.len = b.len + 1;
                                                                    23
       for (int h = 0; h < HA; h++)</pre>
55
                                                                    24
56
            res.hs[h] = ((LL)a * pw[h][b.len] + b.hs[h]) %
                                                                    25
57
       return res;
58
   }
```

```
for (int i = n; i; --i) sa[c[p[i]]--] = i;
    for (int k = 1; k < n; k <<= 1) {
         int cnt = 0;
         for (int i = n - k + 1; i \le n; ++i) t[++cnt] = i; for (int i = 1; i \le n; ++i) if (sa[i] > k) t[+
           \hookrightarrow +cnt] = sa[i] - k;
         for (int i = 1; i \le m; ++i) c[i] = 0;
         for (int i = 1; i <= n; ++i) ++c[p[i]];
         for (int i = 2; i <= m; ++i) c[i] += c[i - 1];
         for (int i = n; i; --i) sa[c[p[t[i]]]--] = t[i],
           \hookrightarrow t[i] = 0;
         swap(p, t);
         p[sa[1]] = cnt = 1;
         for (int i = 2; i <= n; ++i) {
             if (t[sa[i]] != t[sa[i - 1]] || t[sa[i] + k] !=
                \hookrightarrow t[sa[i - 1] + k]) ++cnt;
             p[sa[i]] = cnt;
         if (cnt == n) break;
         m = cnt;
    for (int i = 1; i <= n; i++) rnk[sa[i]] = i;
    for (int i = 1, k = 0; i <= n; i++) {
         if (k) k--;
         while (s[i + k] == s[sa[rnk[i] - 1] + k]) k++;
         height[rnk[i]] = k:
char s[M];
int sa[M], rnk[M], height[M];
int main() {
    cin >> (s + 1);
    int n = strlen(s + 1);
    get_sa(s, n, sa, rnk, height);
    for (int i = 1; i <= n; i++)
         cout << sa[i] << (i < n ? ' ' : '\n');</pre>
    for (int i = 2; i <= n; i++)
         cout << height[i] << (i < n ? ' ' : '\n');</pre>
    return 0;
```

4.7 SAM

```
int lst = 1, cnt = 1, len[M], fa[M], son[M][26];
void Extend(int c) { // 结点数要开成串长的两倍
    int p = lst, np = lst = ++cnt;
   len[np] = len[p] + 1;
    for (; p && !son[p][c]; p = fa[p]) son[p][c] = np;
    if (!p) return fa[lst = np] = 1, void();
    int q = son[p][c];
    if (len[q] == len[p] + 1)
        return fa[lst = np] = q, void();
    int nq = ++cnt;
   len[nq] = len[p] + 1;
   fa[nq] = fa[q];
   fa[np] = fa[q] = nq;
   memcpy(son[nq], son[q], sizeof(son[q]));
    for (; p && son[p][c] == q; p = fa[p]) son[p][c] = nq;
   lst = np:
int c[M], q[M];
int main() {
    for (int i = 1; i <= n ; ++i) Extend(s[i] - 'a');
   for (int i = 1; i <= cnt; i++) ++c[len[i]];
    for (int i = 1; i \le cnt; i++) c[i] += c[i - 1];
   for (int i = 1; i \leftarrow cnt; i++) q[c[len[i]]--] = i;
   return 0;
```

4.8 KMP and EXKMP

```
1 // 1-based
2 int fail[M];
3 void KMP(const char *s, int n) {
4 fail[0] = fail[1] = 0;
5 for (int i = 2, j = 0; i <= n; i++) {
6 fail[i] = 0;
7 while (j && s[i] != s[j + 1]) j = fail[j];
8 if (s[i] == s[j + 1]) fail[i] = ++j;
9 }
10 }
11 // match
```

4.6 SA

```
12
   for (int i = 1, j = 0; i <= la; ++i) {
                                                                           height[tim++] = val[lst];
       while (j && b[j + 1] != a[i]) j = fail[j];
13
                                                                   8
                                                                           sa[tim] = id[x];
14
       if (b[j + 1] == a[i]) ++j;
                                                                   9
                                                                           lst = x;
15
       if (j == lb) {
                                                                        }
           printf("%d\n", i - lb + 1);
                                                                        for (int c = 0; c < 26; ++c)
16
                                                                  11
17
           j = fail[j];
                                                                         | if (son[x][c]) dfs(son[x][c]);
                                                                        lst = fa[x];
18
                                                                  13
19
                                                                  14
   // 0-based
20
                                                                     int main() {
                                                                  15
   // s 和 s 的每一个后缀的最长公共前缀 (LCP) 长度数组
21
                                                                  16
                                                                      | lst = ++cnt;
   void exKMP(const char *s, int *z, int n) {// get z
                                                                  17
                                                                        scanf("%s", s + 1);
23
       int 1 = 0, r = 0;
                                                                  18
                                                                        int n = strlen(s + 1);
24
       z[0] = n;
                                                                        for (int i = n; i; --i) {
25
       for (int i = 1; i <= n; ++i) {
                                                                  20
                                                                           expand(s[i] - 'a');
26
           z[i] = i > r ? 0 : min(r - i + 1, z[i - 1]);
                                                                  21
                                                                         | id[lst] = i;
27
           while (i + z[i] < n \&\& s[z[i]] == s[i + z[i]]) +
                                                                  22
                                                                        }
             23
                                                                        vis[1] = 1;
           if (i + z[i] - 1 > r) r = i + z[l = i] - 1;
28
                                                                        for (int i = 1; i <= cnt; ++i) if (id[i])
                                                                            | for (int x = i,pos = n; x && !vis[x]; x = fa[x]) {
29
                                                                  25
30
                                                                                 vis[x] = 1;
                                                                  26
   // t 与 s 的每一个后缀的 LCP 长度数组
31
                                                                  27
                                                                                  pos -= val[x] - val[fa[x]];
   void exKMP(const char *s, const char *t, int *z, int *p,
32
                                                                  28
                                                                                  son[fa[x]][s[pos + 1] - 'a'] = x;
     \hookrightarrow int sn) {// get p
                                                                              }
                                                                        dfs(1);
33
                                                                  30
       int l = -1, r = -1;
       for (int i = 0; i <= sn; ++i) {
                                                                        for (int i = 1; i <= n; ++i) printf("%d",sa[i]);</pre>
34
           p[i] = i > r ? 0 : min(r - i + 1, z[i - 1]);
35

   puts("");
36
           while (i + p[i] < sn \&\& t[p[i]] == s[i + p[i]]) +
                                                                  32
                                                                        for (int i = 1; i < n; ++i) printf("%d",height[i]);</pre>
                                                                          → puts("");
             → +p[i];
           if (i + p[i] - 1 > r) r = i + p[l = i] - 1;
37
                                                                  33
                                                                        return 0;
38
                                                                  34
39
   }
  4.9 Lydon
                                                                     4.11
                                                                            后缀平衡树
                                                                     const int M=1e5;
   满足s的最小后缀等于s本身的串s称为Lyndon串.
                                                                     bool vis[M << 1];</pre>
```

```
等价于: s是它自己的所有循环移位中唯一最小的一个.
  任意字符串s可以分解为 s = s1s2...sk , 其中 si 是Lyndon串,
   si ≥si+1. 且这种分解方法是唯一的.
6
7
   void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
     for (int i = 0; i < n; ) {
8
9
        int j = i, k = i + 1;
        mn[i] = i;
         for (; k < n \& s[j] <= s[k]; ++ k)
11
12
         | if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
13
          else mn[k] = j = i;
14
        while(i <= j)i += k - j;
15
16
  } // lyn+=s[i..i+kj1]
17
   void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
    | fill(mx, mx + n, -1);
18
19
     for (int i = 0; i < n; ) {
20
        int j = i, k = i + 1;
        if (mx[i] == -1) mx[i] = i;
21
        for (; k < n \&\& s[j] >= s[k]; ++k) {
23
         | j = s[j] == s[k] ? j + 1 : i;
24
           if (mx[k] == -1) mx[k] = i;
        while(i \le j)i += k - j;
26
27
28
  }
```

4.10 SASAM后缀树

```
const int M=1e5;
bool vis[M << 1];
char s[M];
int id[M << 1], ch[M << 1][26], height[M], tim = 0;
void dfs(int x) {
   | if (id[x]) {</pre>
```

```
char s[M];
   int id[M << 1], ch[M << 1][26], height[M], tim = 0;</pre>
   void dfs(int x) {
    | if (id[x]) {
         height[tim++] = val[lst];
 8
         sa[tim] = id[x];
         lst = x;
10
      }
      for (int c = 0; c < 26; ++c)
       | if (son[x][c]) dfs(son[x][c]);
12
13
      lst = fa[x];
14
15
   int main() {
    | lst = ++cnt;
16
17
      scanf("%s", s + 1);
      int n = strlen(s + 1);
18
      for (int i = n; i; --i) {
19
20
         expand(s[i] - 'a');
       | id[lst] = i;
      }
23
      vis[1] = 1;
      for (int i = 1; i <= cnt; ++i) if (id[i])</pre>
24
25
           | for (int x = i,pos = n; x && !vis[x]; x = fa[x]) {
              | vis[x] = 1;
26
27
                pos -= val[x] - val[fa[x]];
                son[fa[x]][s[pos + 1] - 'a'] = x;
28
29
            }
30
      dfs(1);
      for (int i = 1; i <= n; ++i) printf("%d",sa[i]);</pre>
31

    puts("");
      for (int i = 1; i < n; ++i) printf("%d",height[i]);</pre>
        → puts("");
33
      return 0;
34
   }
```