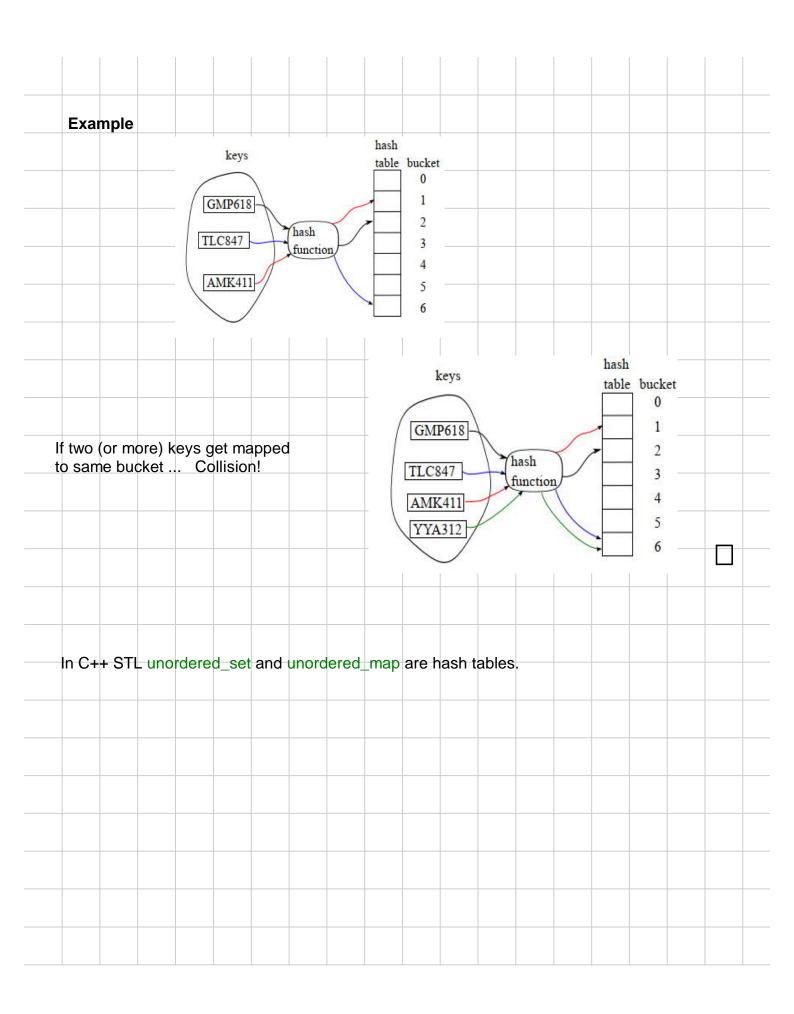
пазн	ables							
	ground and		gy					
	olving collision function	ons						
4. Reh								
	ime efficienc	y and amo	ortized a	analysis				
1. Bacl	ground and	terminolog	ду					
At star	: we have ma	ny pairs: (k	$(1, r_1), (k$	$(k_3, r_2), (k_3, r_3)$	r_3),	(k_n, r_n)		
In each	pair: k_i is uni	que key and	$d r_i$ is a i	record				
Goal	naintain all rec	ords in som	e data st	ructure th	at allows	efficient	insert de	lete a
	operations	ordo in boin	c data bu	ractare en	ar arone	Cincient	msert, de	1000, 0
			ı	1 1	1 1	1 1	1 1	1
Example								
A databa	se of cars							
l 1	cense plate							
	1-5				1			
$r_i = ($	car make, car	nodel, mot	or type,	color, etc.	.)			
Consider	vector (or arra	y): A[1]] A[2]	A[3]		A[m]	E	
	,	.,			Visitorio		b .	
Given inc	lex i , we can ver	y quickly						
• fine	the contents of	A[i] (search))					
• ove	write the conte	ats of $A[i]$ wi	ith NIL (c	lelete)				
A 25114	a record into A	(incort)						

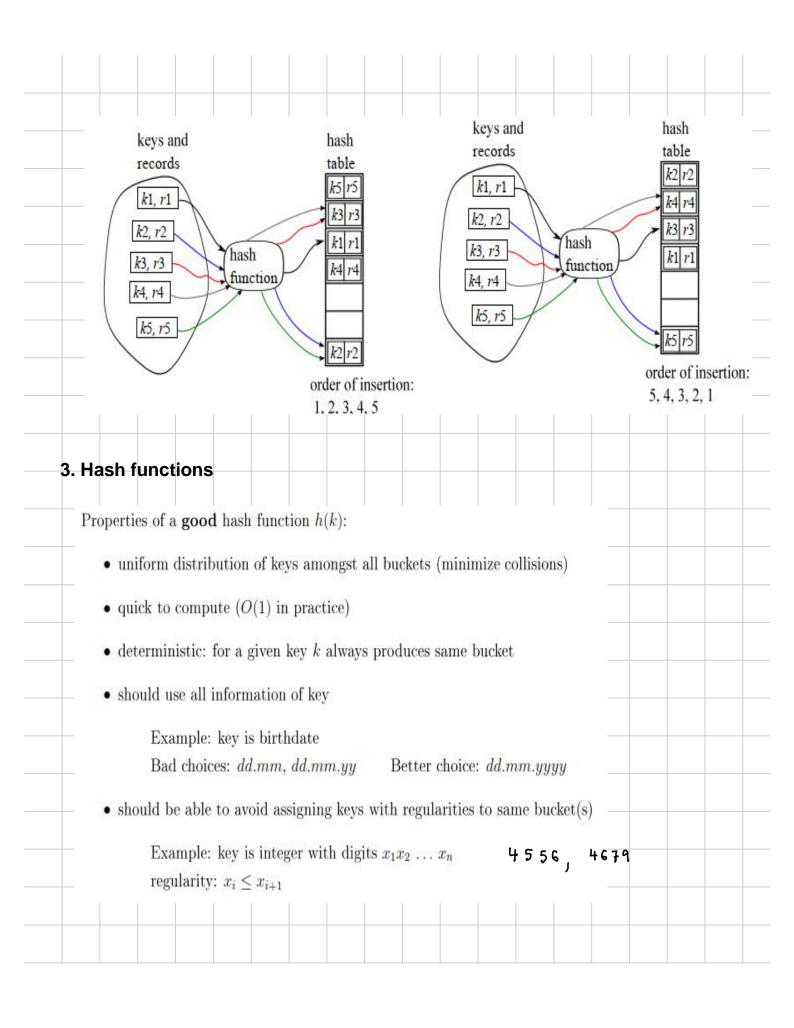
A hash	table	tries	to m	imic	the	efficie	nev o	of an	array	bu:	t only	for	inse	rt	delete	s 5
			(U III	imic	the	Cilicic	ncy c	n an	array	, bu	c Only	101	IIISC	,	ucicu	', '
search	operac	ions.									1					
A 11.	4-11-	· c	1:441-		1	0012362-0012		1		. 1	241		12000		1. 1.	
A hash	table	IS OI	пше	use	wner	1 SOIII	etnin	g na	ving t	o do	with	ora	er is	nee	ded:	
• fi	nd the	large	est or	r sma	allest	key										
• g	iven a	key k	c_i , fin	d the	e nex	t lar	gest o	or the	e next	sma	llest	key				
• g	iven a	key k	i, fin	id if	there	are a	any k	eys 1	arger	than	κ_i					
Townin	logu															
Termino	7.5 (2)=3	34	1			1										
• ke	y: a u	nıque	кеу	assoc	ated	with	a rec	cord								
• re	cord: a	all the	data	a we	want	to st	ore al	bout	one it	em						
• ha	sh tab	le: ar	arra	ay of	buck	ets										
	ish tab						he ha	sh ta	ble							
• bu	icket (or slo	t): 01	ne lo	cation	n in t				npute	s a lo	catio	on (th	e bu	ıcket) i	in
• bu	icket (or slo	t): 01 (or l	ne loc	cation	n in t				npute	s a lo	catio	on (th	e bu	ıcket) i	in
• bu • ha	icket (eash fun ash tab	or slo ction de giv	t): or (or len a	ne loo nash key	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket) i	in
• bu • ha	icket (or slo ction de giv	t): or (or len a	ne loo nash key	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket) i	in
• bu • ha	ash fun ash tab	or slo	t): or (or len a	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket) i	in
buhaha	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket) i	in :
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in :
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in 1
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in i
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in 1
 bu ha co n = num	ash fun ash tab dlision:	or slo	t): or (or len a n two	ne loo nash key o or r	cation map)	n in tl	nctio	n tha	it con				9190 5 3 40 40 9760	e bu	icket)	in 1



```
From https://en.cppreference.com/w/cpp/container/unordered_map
 std::unordered map
    Defined in header <unordered_map>
   template<
        class Key,
        class T,
                                                                                                                  (since
        class Hash = std::hash<Key>,
                                                                                                                  C++11)
        class KeyEqual = std::equal to<Key>,
        class Allocator = std::allocator< std::pair<const Key, T> >
   > class unordered map;
   namespace pmr {
        template <class Key,
                   class T,
                    class Hash = std::hash<Key>,
                                                                                                                  (since
                    class Pred = std::equal_to<Key>>
                                                                                                                  C++17)
        using unordered_map = std::unordered_map<Key, T, Hash, Pred,</pre>
                                      std::pmr::polymorphic_allocator<std::pair<const Key,T>>>;
   }
 Unordered map is an associative container that contains key-value pairs with unique keys. Search, insertion, and
 removal of elements have average constant-time complexity.
 Internally, the elements are not sorted in any particular order, but organized into buckets. Which bucket an element is
 placed into depends entirely on the hash of its key. Keys with the same hash code appear in the same bucket. This
 allows fast access to individual elements, since once the hash is computed, it refers to the exact bucket the element is
 placed into.
  Modifiers
                              clears the contents
   clear(C++11)
                              (public member function)
                              inserts elements or nodes (since C++17)
   insert (C++11)
                              (public member function)
                              inserts an element or assigns to the current element if the key already exists
   insert_or_assign(C++17)
                              (public member function)
                              constructs element in-place
   emplace (C++11)
                              (public member function)
                              constructs elements in-place using a hint
   emplace_hint(C++11)
                              inserts in-place if the key does not exist, does nothing if the key exists
   try emplace (C++17)
                              (public member function)
                              erases elements
   erase (C++11)
                              (public member function)
                              swaps the contents
   swap (C++11)
                              (public member function)
                              extracts nodes from the container
   extract (C++17)
                              (public member function)
                              splices nodes from another container
   merge (C++17)
                              (public member function)
  Lookup
                              access specified element with bounds checking
   at (C++11)
                              (public member function)
                              access or insert specified element
   operator[](C++11)
                              (public member function)
                              returns the number of elements matching specific key
   count (C++11)
                              (public member function)
                              finds element with specific key
   find (C++11)
                              (public member function)
                              checks if the container contains element with specific key
   contains (C++20)
                              (public member function)
                              returns range of elements matching a specific key
   equal range (C++11)
```

2. Resolvi	na colli	sions									
Z. INESOIVI	ing com	SIONS									
Two strat	egies:										
• oper	n hashin	g or chain	ing								
• clos	ed hashi	ng									
Open hash	ning										
Buckets con	tain poi	nters to sta	art of lin	ked list.							
search : T		function g			ucket, if	the rec	ord ex	ists.	We r	nust	sear
search : Tothrough	he hash gh the li he hash gh the li	nked list to	o find the ives the o find the	e key. correct bi e key and	icket, if then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear
search : Tothrough	he hash gh the li he hash gh the li	nked list to function g nked list to	ives the find the	e key. correct be e key and sh	icket, if then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear
search : Tothrough	he hash gh the ling he hash gh the ling and	nked list to function g nked list to	ives the find the	e key. correct but e key and	icket, if then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear
search : To through through through through the search is through through the search is the search i	he hash gh the li he hash gh the li s and rds	nked list to	ives the find the	correct be key and sh	icket, if then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear
keys reco k1, r k4, r4	he hash gh the li he hash gh the li s and rds	nked list to function g nked list to	ives the find the	correct by e key and esh ble NIL k3 r3 NIL NIL NIL	then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear
search: To through through through through the keys reco	he hash gh the li he hash gh the li s and rds	nked list to function g nked list to	ives the find the	correct by e key and esh ble NIL + k3 r3 + NIL + NIL	then de	the recelete the	ord ex (key,	ists.	We n	nust ir.	sear

Bucket in	 terfac	e e										1		1		
begin(si cbegin(s	ze_type)	(C++	11)	returns (public m			the b	eginni	ng of tl	he spe	cified b	oucket				
end(size_ cend(size				returns (public m			the e	nd of t	he spe	cified I	bucket					
bucket_				returns (public m	the nu	ımber d	of bucl	kets								
max_buc	ket_co	unt (C	++11)	returns (public m	the m	aximun	n num	ber of	bucket	S						
bucket_	size(C	++11)		returns (public m			of elen	nents i	n speci	ific bud	cket					
bucket (C++11)			returns (public m			r spec	ific ke	у							
Closed		partition and	1	1	1			1	1	1	1	1			1	
	hash	ing														
			n the	huckets	e											
All data			n the	buckets	S.											
All data	ı is sto	ored in		2216		200.00 TO										
	ı is sto	ored in		2216		red), t	he bi	ucket	from	the ha	ash fu	nction	may	not	be	
All data When o	a is sto	ored in	cur (c	or have	e occu	10110 - 101 0 15 - 10							\$ 1000 O.M.			
All data When of the buc	a is sto	ns ocat	cur (c	or have ur requ	e occu iireme	nts. A	spec	ial pr					\$ 1000 O.M.			
All data When o	a is sto	ns ocat	cur (c	or have ur requ	e occu iireme	nts. A	spec	ial pr					\$ 1000 O.M.			
All data When of the buc with a s	a is sto collision ket the sequen	ns oce at me	cur (cets or	or have ur requ ts that	e occu iireme we sł	nts. A	spec check.	ial pr	obing	functi	ion is	used t	o pro	vide	us	
When of the buc with a sinsert	a is sto collisio ket th sequen : The	ns occat me	cur (ceets or bucke	or have ur requ ts that tion giv	e occu nireme we sh	nts. A nould c	specheck.	ial pr	obing e buck	functi ket is	ion is	used t	o pro	vide n sto	us ore	
When of the buc with a sinsert	a is sto collisio ket th sequen : The	ns occat me	cur (cets or bucke	or have ur requ ts that tion giv	e occu nireme we sh	nts. A nould c	specheck.	ial pr	obing e buck	functi ket is	ion is	used t	o pro	vide n sto	us ore	
When of the buc with a sinsert	a is sto collisio ket th sequen : The	ns occat me ce of hash	cur (cets or bucke funct	or have ur requ ts that tion giv air in i	e occu nireme we sh	nts. A nould c	specheck.	ial pr	obing e buck	functi ket is	ion is	used t	o pro	vide n sto	us ore	
When of the buc with a sinsert	collisions to the collisions that the collisio	ns occat me ce of hash	cur (cets or bucke funct	or have ur requ ts that tion giv air in i	e occu nireme we sh	nts. A nould c	specheck.	ial pr	obing e buck	functi ket is	ion is	used t	o pro	vide n sto	us ore	
When of the buc with a sinsert	collisions to the sequents: The e (key noccup	ns occat me ce of hash , reco	cur (cets or bucke funct rd)-pa ucket.	or have ur requ ts that tion give air in it	e occurireme we shows the	nts. A nould o e a bu he buc	specheck. cket. ket is	ial pr	obing e buck pied, t	functi ket is use pro	not oo	used to	o pro	vide en sto find	us ore an	
When of the buc with a sinsert the unserth	collision ket the sequents: The lecture (key noccur	ns occat me ce of hash , reco	cur (cets or bucke funct rd)-pa ucket.	or have ur requ ts that tion give air in it	e occurireme we shows the t. If the ves a	nts. Anould on the bucket bucket	specheck. cket. ket is	If the occup	obing e buck pied, t	functive function for the function function for the function function for the function functi	not of obing	used to ccupied function key, t	o pro l, the on to hen v	vide en sto find a	us ore an	
When of the buc with a sinsert the unsert for	collision ket the sequent is the constant of the constant is the constant of the constant is the constant in the constant is the constant in t	ns occat me ce of hash pied be hash ne (ke	cur (cets or bucke funct rd)-pa ucket. r func	or have ur reque ts that tion give air in it tion give ord)-pa	e occurireme we shows the vest the vest a ve	nts. Anould on the bucket the bucket	specheck. cket. ket is	If the occup	e buck pied, u acket con	functive function for the function func	not of obing ns the he key	used to	o pro l, the on to hen v	vide en sto find a	us ore an	
When of the buc with a sinsert the unsert for	collision ket the sequent is the constant of the constant is the constant of the constant is the constant in the constant is the constant in t	ns occat me ce of hash pied be hash ne (ke	cur (cets or bucke funct rd)-pa ucket. r func	or have ur requ ts that tion give air in it	e occurireme we shows the vest the vest a ve	nts. Anould on the bucket the bucket	specheck. cket. ket is	If the occup	e buck pied, u acket con	functive function for the function func	not of obing ns the he key	used to ccupied function key, t	o pro l, the on to hen v	vide en sto find a	us ore an	
When of the buc with a sinsert the ur search fo	collision ket the sequent is the constant of t	ns occat me ce of hash ied be hash iee (ke)	cur (cets or bucke funct rd)-pa ucket. func y, reco	or have ur reque ts that tion give air in it tion give ord)-pa	e occurireme we shows the vest the vest a vest a lir. If	nts. Anould on the bucket the bucket ave che	speceheck. cket. ket is tild teket ceket	If the occup	e buck pied, u acket con cupiec	functi ket is use pre contain tain t l buck	not of obing ns the he key sets.	used to ccupied function key, to then	o pro	vide en sto find a ve ha	ore an ve	
When of the buc with a sinsert the ur search fo the delete	collision ket the sequent is The le (key noccupate und the profession of the collision of the collision of the profession of the collision of	ns occat me ce of hash pied by hash hash hash hash	cur (cets or bucke funct rd)-pa ucket. func y, reco	or have ur requests that tion givenir in it tion givenition givenity	e occurireme we shows the ves a wir. If the level a we have a	nts. Anould on the bucket the bucket bucket bucket bucket	speceheck. cket. ket is cket ceket ceket	If the occup the but all occup	e buck pied, u cket con cupied	functi ket is contain tain t buck	not of obing ns the key sets.	used to ccupied function key, to then	o pro l, the on to hen v we m	vide en sto find a ve ha ust u	us ore an ve se	
When of the buc with a sinsert the ur search fo the delete	collision ket the sequent is The le (key noccupate und the profession of the collision of the collision of the profession of the collision of	ns occat me ce of hash pied by hash hash hash hash	cur (cets or bucke funct rd)-pa ucket. func y, reco	or have ur requests that tion givenir in it tion givenition givenity	e occurireme we shows the ves a wir. If the level a we have a	nts. Anould on the bucket the bucket bucket bucket bucket	speceheck. cket. ket is cket ceket ceket	If the occup the but all oc	e buck pied, u cket con cupied	functi ket is contain tain t buck	not of obing ns the key sets.	used to ccupied function key, to then	o pro l, the on to hen v we m	vide en sto find a ve ha ust u	us ore an ve se	
When of the buc with a sinsert the ur search fo the delete for	collision ket the sequent is The le (key moccupate in the le problem in the le probl	ns occat me ce of hash pied be hash hash hash hash hash hash hash has	cur (cets or bucke functed)-paucket. I function function function, reco	or have ur requests that tion givenir in it tion givenition givenity	e occurireme we shows the ves a wir. If the ves a wir. If	nts. Anould on the bucket the bucket bucket the bucket	cket. cket is cket cecked If the	If the occupation of the business occupations occupati	e buck pied, u acket of cupied cket of ot con	function function function the function the function the function	not of obing the key sets. In the key he key he key	used to ccupied function key, to then	o pro l, the on to hen v we m	vide en sto find a ve ha ust u	us ore an ve se	



Exa	mple																
Δο	sume	kev	l ie i	nteg	er en	d m	ie nu	ımbei	of h	neke	te						
710	sume	Key	n 10 1	псев	ci an	id m	15 110	unoci	OI U	ucke	05						
C:.	nnlo	book	f	tion.	1./1.)		1/1						1/-				
511	nple	nasn	Tune	tion;	$n(\kappa)$	= n	A)DOI	(c, m)				m	9 q (3	7,5)	۶ <u>ک</u>		
_ w	hen	migh	t this	h(k) he	had?											
,,		15.70		22 2													
		100100					. 1	are ev			i						
	C	onsec	luenc	e: na	ali oi	buck	cets a	are al	ways	emp	ty						
	• St	ippo	se m	= 10	ane	d key	is ir	ntege	r wit	h s >	$r ext{ di}$	gits			h :	: 10(2 A
							1	1.0	v0 ·	101	II.		10				
							$\kappa =$	$\alpha_0 10$	+ c	$\chi_1 \perp U^*$	+ .	+	$\alpha_s 10$		k =	764	87
															••		
	C	onsec	iuenc	e: da	ata o	$_{r}$, α_{r}	_11 , .	α _s	is no	t use	ed				•		
	c	onsec	luenc	e: da	ata α	$_r, \alpha_r$	+1, •	α_s	is no	t use	ed						
	C	onsec	luenc	e: da	ata α	$_r$, α_r	+1, ·	α _s	is no	t use	ed						
	C	onsec	luenc	e: da	ata α	r , α_r	+1, •	α_s	is no	ot use	ed						
	C	onsec	luenc	e: da	ata α	r, α_r	+1, •	$$ α_s	is no	t use	ed						
	ible p											ocus	on p	rovic	ling g		
Sens	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovic			
	ible p											ocus	on p	rovio			
	ible p											ocus	on p	rovic			

				0.71													
Hash	n tunc	ctions	and	SIL:													
	doole	, with a															
	decla	9201													<u>~</u> }		
1	ا ـــــــــ	std::u	norder	red_ma	p <std:< td=""><td>::stri</td><td>ng, my</td><td>/Data></td><td>myMap</td><td>);</td><td></td><td></td><td></td><td></td><td>_</td><td></td><td></td></std:<>	::stri	ng, my	/Data>	myMap);					_		
	simr	olv use	s STL	's defa	ult ha	sh fur	nction	when	kev is	string	ř						
												2000 - 2000			-0.000		
						7 2500) or po	ointer	or stri	ng, be	st to u	se ST1	L's		
	own	derau	it nasn	and 1	ocus o	on tori	ming k	eys									
	unor	dered	l_map	will al	low us	er to	input (own ha	ash fur	nction							
	if ke	y is st	ruct,	it mig	ght be	possil	ole to 1	ise soi	ne att	ribute	as ke	y					
1	1 .	struct	Car	E 337		17.0						2					
	2 -	{	oar												_		
3	3	int	yearMa	ide;													
4	4			1000		ate;	// Bes	t pos	sible	key.							
35	5		:strin														
	3 7	std: };	:strin	ig Mot	orType	;											
8	_	,													-		
STL w	vill no	t acce	ept a	struct	as a	key.	In su	ch ca	ses it	may	be ne	ecess	ary to	form	a has	sh fur	oction
Two e	examp	oles c	f form	ning a	hash	func	tion fi	om a	struc	t.							
1																	

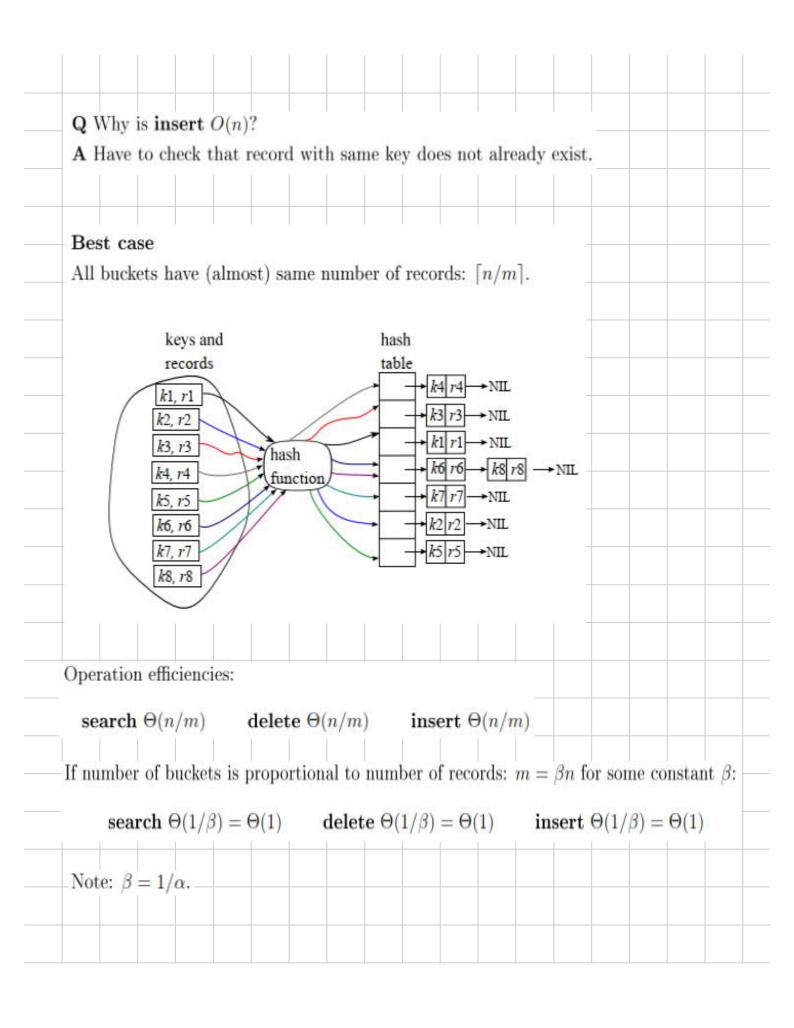
```
Example I
      struct Key {
  2
        std::string first;
  3
        std::string second;
  4
      };
  5
  6
     struct KeyHash {
  7
        std::size_t operator()(const Key& k) const
  8
  9
         return std::hash<std::string>()(k.first) ^
                                                                 \wedge
 10
          (std::hash<std::string>()(k.second) << 1);
                                                                 XOR
       }
 11
 12
      };
 13
 14
      struct KeyEqual {
       bool operator()(const Key& lhs, const Key& rhs) const
 15
 16
 17
         return lhs.first == rhs.first && lhs.second == rhs.second;
        }
 18
 19
      };
 20
 21
      int main()
 22
 23
        // Define the KeyHash and KeyEqual structs and use them in the template
        std::unordered_map<Key, std::string, KeyHash, KeyEqual> newMap = {
 24
 25
          { {"John", "Doe"}, "example"}, { {"Mary", "Sue"}, "another"} };
 26
 27
      }
```

	Example	e II													
	//														
	// Type		oordinat	te (x,	У										
	{														
4	int	x = NO_V	/ALUE;												
5	int	y = NO_V	/ALUE;												
6	};														
7															
0.00	// Examp		The state of the s						so tha	t it	can be	e used			
1,000	// as ke	The second second second			DESCRIPTION OF SERVICE										
	inline h	_													
- 11	inline h	and the same of th	rator!=	(Coord	c1, Coo	ord c2)	{retu	ırn !	(c1==c	2); }	// No	ot str	ictly		
12		ssary													
0.000	struct ('oordHag	h												
14		Joorunas	11												
15		:size_t	operato	r()(Coc	ord xv)	const									
16		.0120_0	oporado	() (00	, L u, ,									_	
17		auto has	her = s	td::has	h <int></int>	();									
18	<u></u>	auto xha	sh = has	sher(xy	.x);										
19	-	auto yha	sh = has	sher(xy	.y);										
20		// Combi	ne hash	values	(magi	c!)									
21		return x	hash ^	(yhash	+ 0x9e	3779ъ9	+ (xha	ash <	< 6) +	(xha	sh >>	2));			
22	33														
23	};														
														П	
If vo	ur key is	really	compo	sed of	two	r mor	e attri	ihute	s (el	emen	te) t	hen s	earch	ı (Go	oald
	elp to o						o atti	ibato	0 (01		ιο,, τ		ou. o.	. (00	~ <u>9</u>
	•														

	Rehas	hing												
Q:	What	is a ha	ash tab	le's loa	nd facto	or α ?								
A :	The lo	ad fac	tor is t	he ave	rage nu	ımber	of reco	rds per	buck	et or o	$\alpha = r$	n/m.		
Q:	What	is a d	one in	rehashi	ing?									
100			vo part		1670								_	
		-2008 A. L. C. 190	1.1 1.	m • (0 esweed a	5.0000000000000000000000000000000000000	e (1	1 1	e .						
	• an	accept	able lin	art α_{lin}	ı is set	for th	e load	tactor						
	• if a	$\alpha > \alpha_{lin}$	n, then	a new	hash t	able is	create	d with	m_{new}	> m 1	oucke	ts		
-	- 11	1	c	111	1 / 11		1 1	1 1	7	1 .	1	1 1		
	• an	record	s irom	oid nas	sn table	e are r	enasneo	l and r	estore	1 in ne	ew na	asn ta	oie	
Dec	cisions:													
1000100														
1	valu	e for α	lim?											
	• valu	e for n	lnow?											
			new.											
TI	ne valu	e for α	lim is a	compr	omise:		-							
+			cent.											
	6.00 0000	Cathalian State of the La					See the second see	needed	, but r	nore r	ecord	s per l	oucket	an
	hen	ice long	ger avei	age se	arch a	nd del	ete tim	ies						
	• sma	all α _{lim}	: more	bucket	s and h	ence n	nemorv	needed	, but f	ewer r	ecord	s per l	oucket	ar
			rter ave				same of		,			1		
	5303,074.0		70 70 00				03-9/00/53/00/53	C C 234-94						

- Ch	oosir	or vol	luo fo	r rot	io m	10	n ic	a com	Drom	ico.							
CII	100511	ig vai	iue io	ı ıaı	10 111	new/	11 15	a com	ргоп	nse.							
	• sn	naller	m _{net}	w/m:	mo	re fre	quen	t reha	ashes.	, less	men	nory	used	whe	n rel	ashi	ng
	• la	rger i	m_{new}	/m:	less f	freque	ent re	ehash	es, m	ore i	neme	ory u	sed v	vhen	reha	shing	g
		553	20			50						52					
STL	's unc	order	ed se	et and	d unc	order	ed ma	ap hav	re aut	omat	ic rel	ashir	19° 11Si	ng de	efault	valu	es
		$d m_{ne}$					- u	-р					.0			, ara	
	CHIL COLL	- ····	w.														
Howe	ever, i	ıser c	an se	t the	se.												
						onco	COM	/u/cr	on/co	nta.	inor	/1120	rdor	od m	an		
						ence	.com	/w/cp	op/co	onta:	iner	/uno:	rder	ed_m	ap		
Fron		ps:/		cppr	efer				19509			/uno:	rder	ed_m	ар		
Fron Hash	n htt policy	ps:/	/en.	cppro	efer	erage r	number	of elen	nents p	er buc	ket			ed_m	ар		
Fron Hash load	n htt policy	ps:/	/en.	retu (publ man	efer	erage r ber funct maximu ber funct	number ion) um ave ion)	r of elem	nents p	er buc	ket ients p	er bucl	cet				
Fron Hash load max	n htt policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct	number ion) im ave ion) the spe	r of elem rage nu ecified n	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		
Fron Hash load max_ reha	n htt policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		nsh
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		ish
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		ısh
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		ash
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o number	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		nsh
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		nsh
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		ish
Fron Hash load max_ reha	policy	ps:/	/en.	retu (publ man (publ rese (publ	efer	erage r ber funct maximu ber funct t least ber funct pace fo	number ion) um ave ion) the spe ion) r at lea	r of elem rage nu ecified n	nents p ımber o	er buc of elem	ket nents p	er buck	cet enerate	es the I	hash ta		nsh

100		g (open h	ashing)	zed analysi				
Worst		0.00						
_	keys and	i		hash table				
	k1, r1 k2, r2 k3, r3 k4, r4	has	h		3 → k4 r4	→ k1 r1 -	→[k5]r5]	k2 r2 —



Two	o requ	ireme	ents t	o ob	tain 1	runtii	me ef	ficien	cies t	hat a	are co	onsta	nt ti	me:			
	• <i>m</i> i	s pro	porti	onal	to n												
-	• god	od ha	ash f	unct	ion!	(doe	s not	have	to b	e per	fect)						
				2000 mg p 16 m Coll 70 1	200, 780 5 0-0. 0				1 - 2000 0 500 2 400								
0.	Wha	t to c	lo wl	on n	gros	we?											
	Reha		10 W1	icii n	gro	ws.											
54 2	7000	250	039	£36399	204 0		33	33	200 8								
	Do w			17.4					shing	?							
A :	Not i	we c	consic	ier ai	norti	zed a	nalys	sis.									
, 34 .	What						W/S	m							e		5/45
A :	Instea	d of J	ust co	onside				ne effi	cienc	y for			W. 72 P. S. C. C. C.		dure	or op	eration
THO	concid	lor th	0 0770	POGO	minti	maa	Hiciar	our fo	n o c.	omon	no of	OTOM	entin	CT CT			
we	consid	ler th	e ave	rage	runti	me e	fficier	icy fo	r a se	equen	ce of	ope	ratio	ns.			
			e ave	rage	runti	me e	fficier	icy fo	r a se	equen	ce of	ope	ratio	ns.			
	consid		e ave	rage	runti	me e	fficier	ncy fo	r a se	equen	ce of	oper	ratio	ns.			
Exa			e ave	rage	runti	me e	fficier	icy fo	r a se	equen	ce of	ope	ratio	ns.			
Exa	ample sumpt	ions:		****				17					ratio	ns.			
Exa Ass	ample sumpti perfor	ons: m a se	equeno	****				17					ratio	ns.			
Exa Ass	ample sumpt	ons: m a se	equeno	****				17					ratio	ns.			
Ass	ample sumpti perfor	ons: m a se t is Θ(equeno(1)	ce of i	nsert	opera	ations	and r					ratio	ns.			
Ass	ample sumpti perfor insert	ons: m a se t is Θ(equence(1)	ce of i	nsert l facto	operator is α	ations	and r	ehasł	ı oper	ations	8			(n)		
Ass	ample sumpti perfor	ons: m a se t is Θ(equence(1)	ce of i	nsert l facto	operator is α	ations	and r	ehasł	ı oper	ations	8			(n)		
Ass	ample sumpti perfor insert	ons: m a se t is Θ(equence(1)	ce of i	nsert l facto	operator is α	ations	and r	ehasł	ı oper	ations	8			(n)		
Ass	ample sumpti perfor insert	ons: m a se t is Θ(equence(1)	ce of i	nsert l facto	operator is α	ations	and r	ehasł	ı oper	ations	8			(n)		
Ass	ample sumpti perfor insert	ons: m a se t is Θ(equence(1)	ce of i	nsert l facto	operator is α	ations	and r	ehasł	ı oper	ations	8			(n)		

Operation sequence	e:		
stage 1 Set has	n table size to $m_0 = 1$. I	Do one insert . Set $i = 1$.	
stage 2 Set new	hash table size to $m_i =$	$2m_{i-1}$. Perform a rehash .	
stage 3 Do m_i	$-m_{i-1}$ insert operations	s. Set $i = i + 1$.	
		n repeat stages 2 and 3.	
stage 4 II t = t.	then stop. If $i < r$, then	ii repeat stages 2 and 5.	
$i \mid m_{i-1} \mid m_i$	stage 2 operations count	ts stage 3 operations counts	
$-\begin{array}{c ccccccccccccccccccccccccccccccccccc$	1	1	
- 2 2 4	2	2	
- 3 4 8	4	4	
- : :			
	5032 x 011	where •	
$r \mid 2^{r-1} \mid 2^r$	2^{r-1}	2^{r-1}	
Total number of el	ements in hash table:		
	n = stage 1 insert + st	90 D) 900 P	
	= 1 + 1 + 2 + 4 + .	$\ldots + 2^{r-1} = 2^r$	

Total operatio	ns count:			
T -	stage 1 count	cum of stage 2 coun	ts + sum of stage 3 cou	nte
	$1 + 2^r - 1 + 2^r - 1$		ts + sum of stage 5 cou	III.S
Amortized ef	ficiency per operation	on:		
	Barrestonium and attachman in the			
	$\frac{T}{n} = \frac{2^{r+1}-1}{2^r}$	$\leq 2 = O(1)$		
— Critical part	: in stage 2, new ha	sh table size is $m_i = 1$	fm_{i-1} for some $f > 1$.	

Tämä teos on lis	ensoitu Creati	ive Commo	ns Nimeä-E	iKaupallineı	ղ-		
EiMuutoksia 4.0				le lisenssiä	osoitteess	a	
http://creativecor	HITIOHS.OLG/IICE	enses/by-n	<u>5-110/4.0/</u>				
tekijä: Frank Car	meron						
This work is lice							
NonCommercial this license, visit							
made by Frank (Cameron						
	80						
(cc)(1)	\$=						
© BY	S =						
(C) (B)	S =						
(C) (B) (B)	NC ND						
(C) (BY	NC ND						
© (1) BY	NC ND						
© BY	NC ND						
© BY	S =						
© BY	S =						
GO BY	S = NC ND						
ВУ	S = NC ND						
BY	S = NC ND						
ВУ	S = NC ND						
ВУ	S = NC ND						
ВУ	S = NC ND						
BY	S E NC ND						
BY	S = NC ND						