

Gestor de Cursos y Rutas Académicas.

Dany Bañol Osorio.
Julián Bonilla Mórtigo
José Luis Rativa Medina

Equipo G

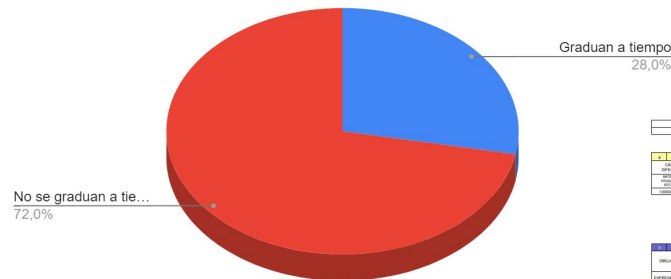


UNIVERSIDAD
NACIONAL
DE COLOMBIA
SEDE BOGOTÁ



Problema que se resolvió

Proporcion de estudiantes que se graduan a tiempo



UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ
PROGRAMA CURRICULAR
PLAN DE ESTUDIOS
INGENIERIA MECATRONICA

Agosto 18 de 2014 de Consejo de Facultad de Ingeniería

I		II		III		IV		V		VI		VII		VIII		IX		X	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420
421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440
441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460
461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480
481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520
521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540
541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560
561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580
581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620
621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640
641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660
661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680
681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720
721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740
741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760
761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780
781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820
821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840
841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860
861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880
881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920
921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940
941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960
961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980
981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000



Mantenimiento del servicio

En estos momentos se están realizando tareas de mantenimiento por lo que el servicio no se encuentra operativo.

Por favor inténtelo más tarde o, en caso de persistir el problema, diríjase al administrador

Rogamos disculpen las molestias que esta actuación pueda ocasionar.





Requerimientos funcionales

Menú de la Consola del prototipo inicial:

- 1) Organizar Asignaturas y Ruta Académica:

```
Bienvenido al gestor de rutas academicas
Opciones:
1. Organizar Asignaturas
2. Selecciona asignaturas para generar rutas
3. Buscar asignatura
4. Cargar un archivo txt para ver mis asignaturas
5. Consultar asignaturas disponibles
Que deseas hacer :
```

```
¿Cuántas asignaturas quieres planificar ?
2
Nombre de la asignatura:
P00
Nombre de prerequisitos(separados por comas):
PB
Nombre de la asignatura:
PB
Nombre de prerequisitos(separados por comas):

Introduzca la ruta del directorio donde quiere que se guarde su archivo:
C:\Users\josel\OneDrive\Documentos\Ing.Sis\2021-2\DataStructures\Proyecto\Prototipo2
Introduzca el nombre para su archivo:
Prueba
5
Semestre 1 --- PB
Semestre 2 --- P00
Presione 1 para volver al menu de opciones, 2 para salir del programa
```



Requerimientos funcionales

2) Seleccionar Asignaturas para generar rutas:

```
2
Introduzca el nombre de la asignatura que quiere cursar:
Fisica computacional
Si quiere cursar "Física computacional"le recomendamos estudiar el programa: 2418
```

3) Buscar Asignaturas:

```
Que desea hacer :
3
Introduzca el nombre de la asignatura que quiere buscar:
Fisica computacional
Codigo: 2027632
Nombre: "Física computacional"
Creditos: 3
Componente: "Formación Disciplinar o Profesional"
Presione 1 para volver al menu de opciones, 2 para salir del programa
```

4) Cargar archivo para ver asignaturas

```
4
Introduzca la ruta del directorio donde está su archivo:
C:\Users\josel\OneDrive\Documentos\dumps
Nombre de su archivo:
Prueba
4
Semestre 1 --- CALCULO DIFERENCIAL
Semestre 2 --- CALCULO INTEGRAL
Semestre 3 --- CALCULO VECTORIAL
Presione 1 para volver al menu de opciones, 2 para salir del programa
```

5) Consultar Asignaturas Disponibles:

```
5
[
{
  "codigo": 2015168,
  "nombre": "Fundamentos de matemáticas",
  "creditos": 4,
  "obligatoria": true,
  "prerrequisito": null,
  "correquisito": null,
  "componente": "Fundamentación",
  "programas": [2418]
},
{
  "codigo": 2015181,
  "nombre": "Sistemas numéricos",
  "creditos": 4,
  "obligatoria": true,
  "prerrequisito": "2015168",
  "correquisito": null,
  "componente": "Fundamentación",
  "programas": [2418]
},
{
```

Implementación de nuevas estructuras de datos en la solución del problema planteado

Listas enlazadas

- Almacenamiento



implementa a



Tablas Hash

- Almacenamiento



Prerrequisitos

Condición 1 Tipo M ¿Todas? [N] Número asignaturas [1]

1000005-B Cálculo Integral

2000916 Matemáticas ii

Condición 2 Tipo M ¿Todas? [N] Número asignaturas [1]

1000003-B Álgebra Lineal

2000916 Matemáticas ii

Implementación de nuevas estructuras de datos en la solución del problema planteado

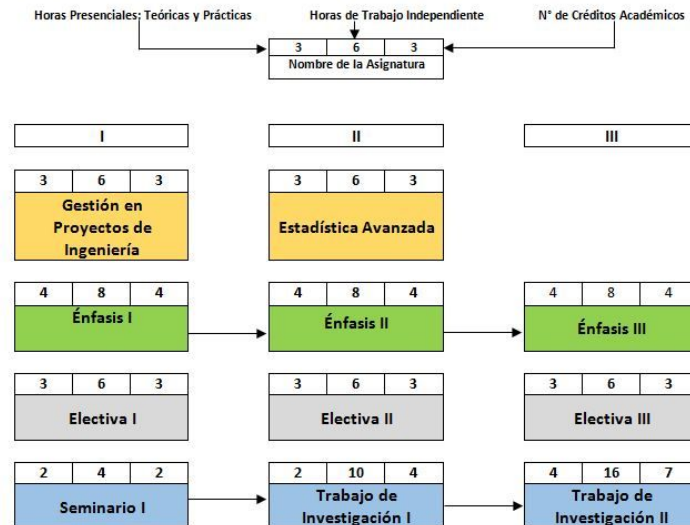
Pila

- Lectura de archivo



Árbol AVL

- Asignación de semestre para asignaturas



Análisis comparativo del uso de las nuevas estructuras de datos implementadas (Gráficas)

Busqueda(asignaturas)

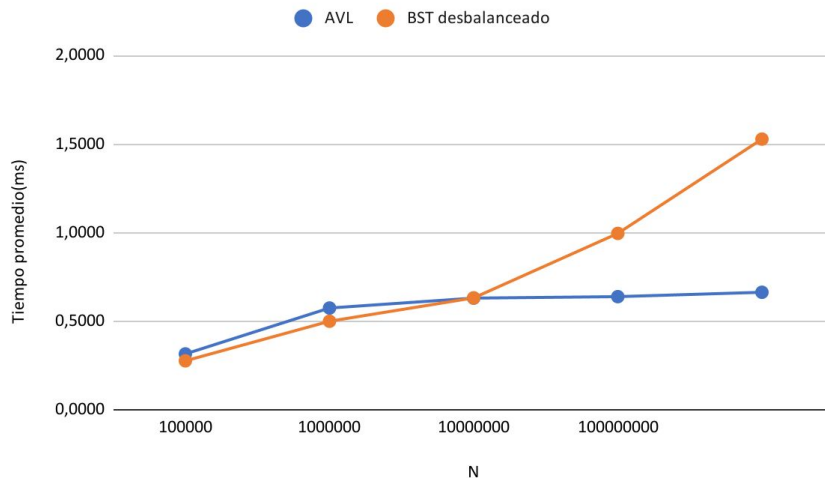


Gráfico 1. Tiempos de ejecución método “buscar” en árbol AVL y BST desbalanceado.

Busqueda (asignaturas) en lista enlazada

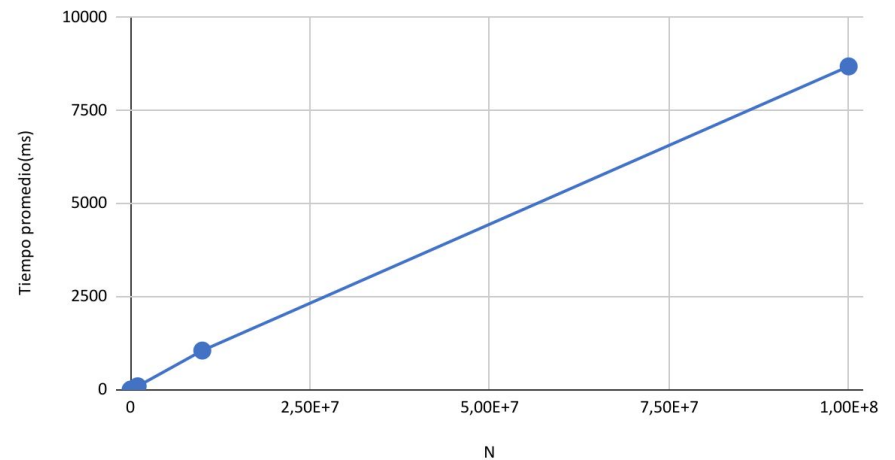


Gráfico 2. Tiempos de ejecución método “buscar” en una lista enlazada.

De la gráfica 1 y 2, podemos concluir que, con el método “buscar” un dato, el crecimiento en el árbol AVL tiende a comportarse de manera logarítmica, mientras que en el árbol BST tiende a ser lineal, al igual que en la lista enlazada .

Análisis comparativo del uso de las nuevas estructuras de datos implementadas (Gráficas)

Eliminación de asignaturas

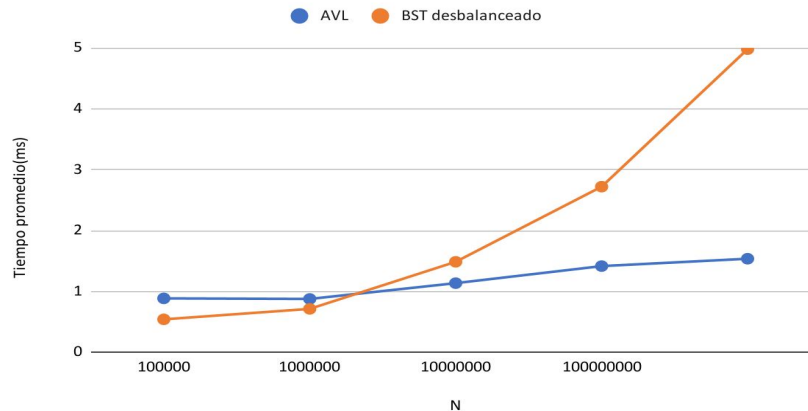


Gráfico 3. Tiempos de ejecución método “eliminar” en árbol AVL y BST desbalanceado.

Eliminación de asignatura en Lista enlazada

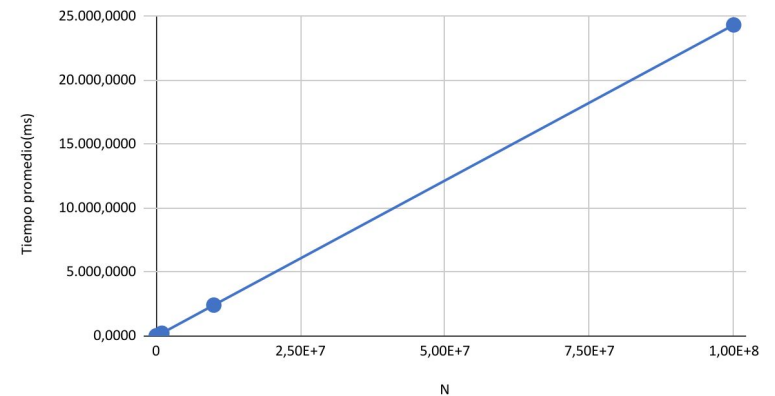


Gráfico 4. Tiempos de ejecución método “eliminar” en Listas enlazadas.

De la gráfica 3 y 4, podemos concluir que, con el método “eliminar” un dato, el crecimiento en el árbol AVL tiende a comportarse de manera logarítmica, mientras que en el árbol BST y la lista enlazada tiende a ser lineal.

Pruebas del uso de las nuevas estructuras de datos implementadas

Se compararon los árboles entre ellos y con la lista enlazada, la cual ya se tenía previamente para saber qué estructura de datos usar en el proyecto.

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Búsqueda de datos (asignaturas)	Árbol AVL	10000	$O(\log n)$	0,3167
		100000	$O(\log n)$	0,5758
		1000000	$O(\log n)$	0,6317
		10000000	$O(\log n)$	0,6400
		100000000	$O(\log n)$	0,6646
	Árbol BST	10000	$O(n)$	0,2777
		100000	$O(n)$	0,5012
		1000000	$O(n)$	0,6329
		10000000	$O(n)$	0,9968
		100000000	$O(n)$	1,5289
	Lista enlazada	10000	$O(n)$	1,189
		100000	$O(n)$	12,784
		1000000	$O(n)$	96,919
		10000000	$O(n)$	1056,623
		100000000	$O(n)$	8689,08

Tabla 1. Tiempos de ejecución del método “Buscar” en Árbol AVL, Árbol BST, Lista enlazada.

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Eliminación de datos (asignaturas)	Árbol AVL	10000	$O(\log n)$	0,8866
		100000	$O(\log n)$	0,8771
		1000000	$O(\log n)$	1,1366
		10000000	$O(\log n)$	1,4167
		100000000	$O(\log n)$	1,5377
	Árbol BST	10000	$O(n)$	0,5421
		100000	$O(n)$	0,7140
		1000000	$O(n)$	1,4872
		10000000	$O(n)$	2,7216
		100000000	$O(n)$	4,9806
	Lista enlazada	10000	$O(n)$	2,7263
		100000	$O(n)$	29,3203
		1000000	$O(n)$	222,2903
		10000000	$O(n)$	2.423,4480
		100000000	$O(n)$	24.339,1731

Tabla 2. Tiempos de ejecución del método “Eliminar” en Árbol AVL, Árbol BST, Lista enlazada.

Para búsqueda el mejor es el árbol AVL

Para eliminación el mejor es el árbol AVL

Pruebas y análisis comparativo del uso de las nuevas estructuras de datos implementadas- Árboles.

Inserción de datos en AVL y BST desbalanceado

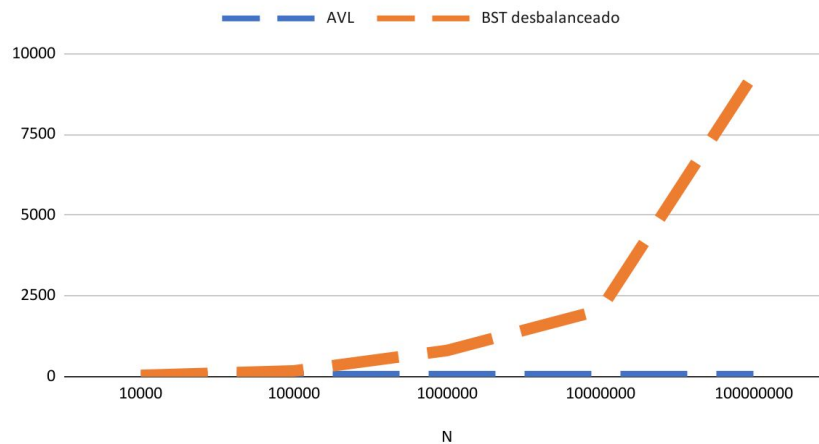


Gráfico 5. Tiempos de ejecución método “insertar” en árbol AVL y BST desbalanceado.

Inserción (asignaturas) en Lista enlazada

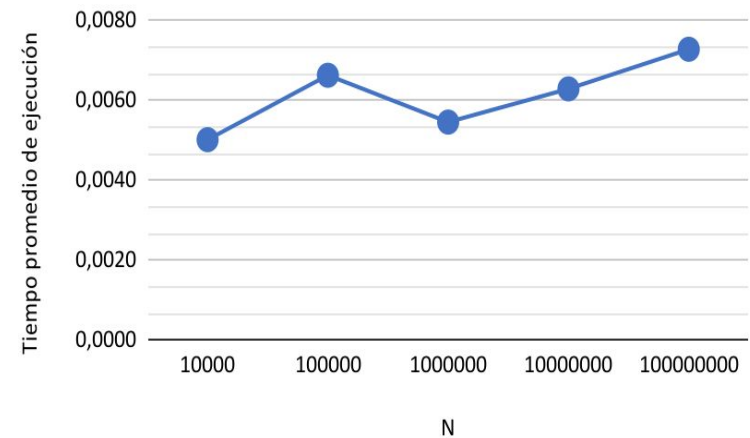


Gráfico 6. Tiempos de ejecución método “insertar” en Listas enlazadas.

De la gráfica 5 y 6, podemos concluir que, con el método “insertar” un dato, el crecimiento en el árbol AVL tiende a comportarse de manera logarítmica, mientras que en el árbol BST tiende a ser lineal y en la lista enlazada se comporta de manera constante.

Pruebas del uso de las nuevas estructuras de datos implementadas

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Inserción de datos (Asignaturas)	Árbol AVL	10000	$O(\log n)$	0,0242
		100000	$O(\log n)$	0,0342
		1000000	$O(\log n)$	0,0401
		10000000	$O(\log n)$	0,0441
		100000000	$O(\log n)$	0,0524
	Árbol BST	10000	$O(n)$	39,3727
		100000	$O(n)$	178,7520
		1000000	$O(n)$	811,5341
		10000000	$O(n)$	2061,2967
		100000000	$O(n)$	9358,2871
	Lista enlazada	10000	$O(1)$	0,0050
		100000	$O(1)$	0,0066
		1000000	$O(1)$	0,0055
		10000000	$O(1)$	0,0063
		100000000	$O(1)$	0,0073

Tabla 3. Tiempos de ejecución del método "Insertar" en Árbol AVL, Árbol BST, Lista enlazada.

Para inserción el mejor es la lista enlazada

Análisis comparativo de las Gráficas con respecto a la Complejidad Temporal

	Buscar Dato (Asignaturas)	Eliminar Dato (Asignaturas)	Insertar Datos (Asignaturas)
Árbol AVL	$O(\log n)$	$O(\log n)$	$O(\log n)$
Árbol BST Desbalanceado	$O(n)$	$O(n)$	$O(n)$
Lista Linkeada	$O(n)$	$O(n)$	$O(1)$

Ya que el árbol AVL es mejor que el árbol desbalanceado y la lista enlazada en búsqueda y eliminación de datos, consideramos al árbol AVL como una mejor opción para almacenamiento de datos.

Pruebas y análisis comparativo del uso de las nuevas estructuras de datos implementadas- Árboles.

Desencolamiento en max heap

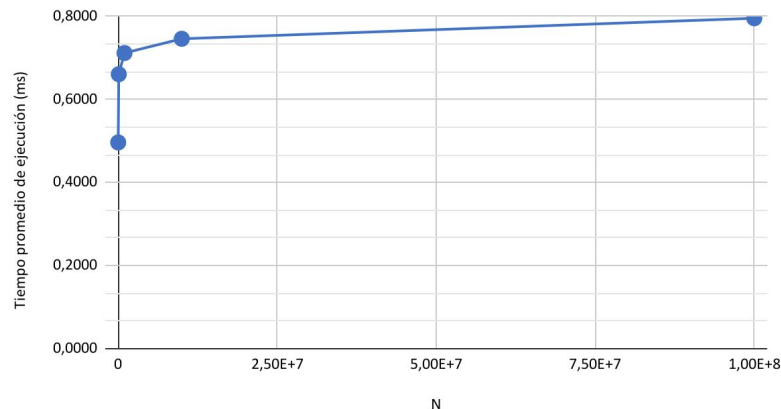


Gráfico 7 . Tiempos de ejecución método “desencolar” en max heap.

Desencolamiento en cola de prioridad implementada con lista enlazada

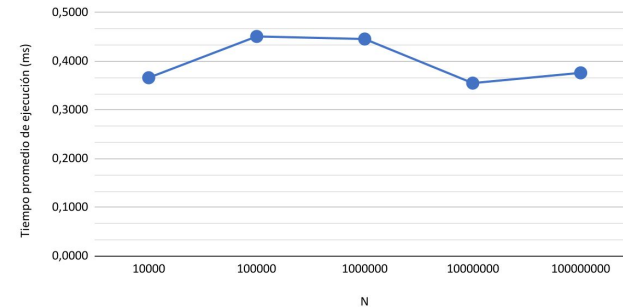


Gráfico 8. Tiempos de ejecución método “desencolar” en cola de prioridad implementadas con Listas enlazadas.

De la gráfica 7 y 8, podemos concluir que, con el método “Desencolar” un dato, el crecimiento en el Max Heap tiende a comportarse de manera logarítmica, mientras que en la cola de prioridad tiende a ser constante .

Pruebas y análisis comparativo del uso de las nuevas estructuras de datos implementadas- Heaps.

Encolamiento de valor con menor prioridad a los que estan en cola (maxHeap)

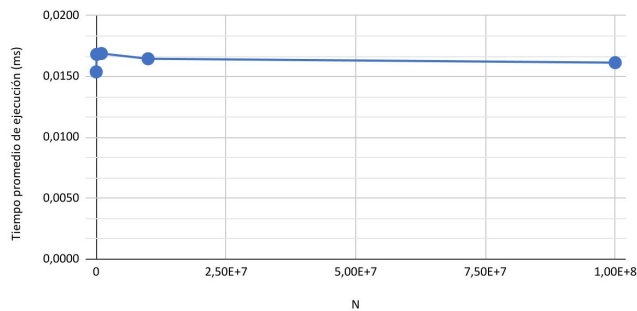


Gráfico 9 . Tiempos de ejecución método “encolar” con menor prioridad a los que están en cola (max heap).

Encolamiento de valor con menor prioridad a los que estan en cola - Lista Enlazada

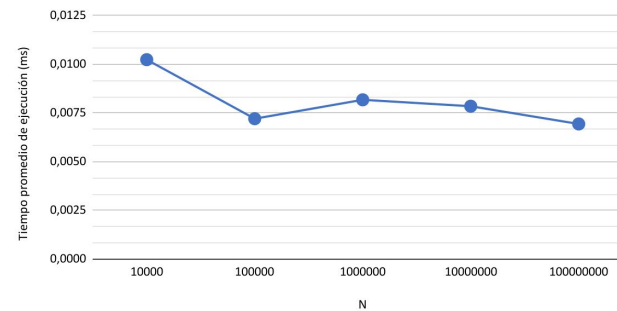


Gráfico 10. Tiempos de ejecución método “encolar” con menor prioridad a los que están en cola de prioridad implementadas con Listas enlazadas.

Encolamiento de valor con mayor prioridad a los que estan en cola (maxHeap)

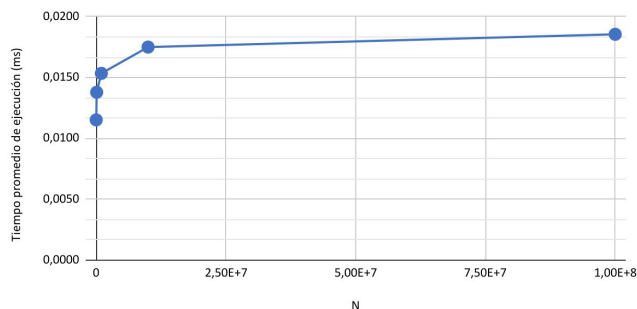


Gráfico 11 . Tiempos de ejecución método “encolar” con mayor prioridad a los que están en cola (max heap).

Encolamiento de valor con mayor prioridad a los que estan en cola - Lista Enlazada

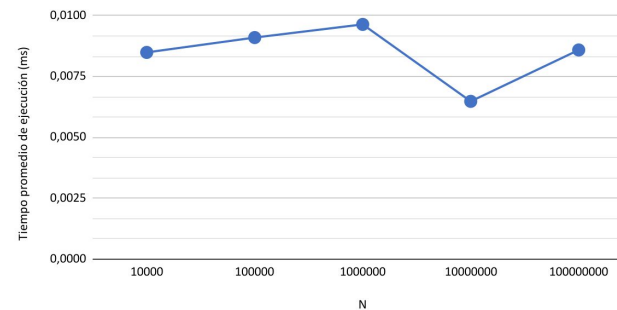


Gráfico 12. Tiempos de ejecución método “encolar” con mayor prioridad a los que están en cola de prioridad implementadas con Listas enlazadas.

Pruebas y análisis comparativo del uso de las nuevas estructuras de datos implementadas- Heaps.

Encolamiento de valor con prioridad media con respecto a los que están en cola - Max Heap

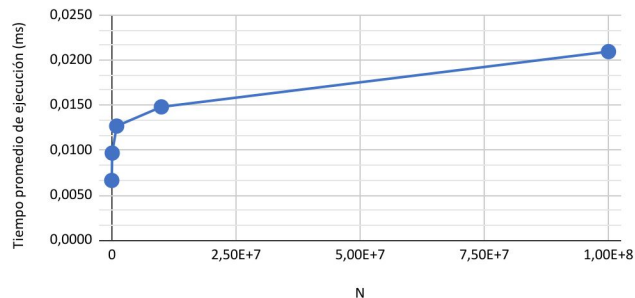


Gráfico 13 . Tiempos de ejecución método “encolar” con valor de prioridad media con respecto a los que están en cola (max heap).

Encolamiento de valor con prioridad media con respecto a los que están en cola - Lista Enlazada

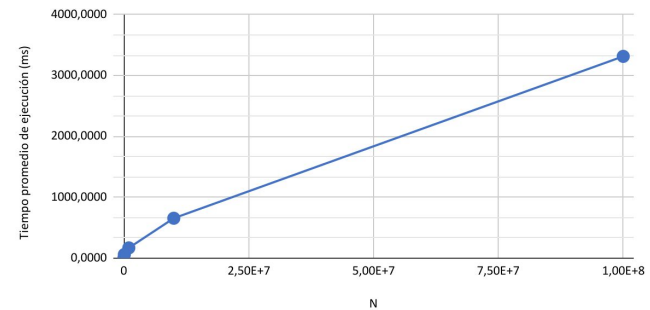


Gráfico 14. Tiempos de ejecución método “encolar” con valor de prioridad medio con respecto a los que están en cola de prioridad implementadas con Listas enlazadas.

De la gráfica 9 y 10, podemos concluir que, con el método “encolar valor con menor prioridad”, el crecimiento en el Max Heap tiende a comportarse de manera constante al igual que en la cola de prioridad.

De las gráficas 11 y 12, podemos concluir, con el método “encolar valor con mayor prioridad”, el crecimiento en el Max Heap tiende a comportarse de manera logarítmica, mientras que en la cola de prioridad tiende a ser lineal.

De las Gráficas 13 y 14, podemos decir que el crecimiento en el Max Heap tiende a ser logarítmico, mientras que en la cola de prioridad, tiende a ser lineal.

Pruebas del uso de las nuevas estructuras de datos implementadas

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Desencolamiento	Max Heap	10000	$O(\log n)$	0,4965
		100000	$O(\log n)$	0,6608
		1000000	$O(\log n)$	0,7119
		10000000	$O(\log n)$	0,7461
		100000000	$O(\log n)$	0,7955
	Cola de prioridad basada en Lista enlazada	10000	$O(1)$	0,3664
		100000	$O(1)$	0,4511
		1000000	$O(1)$	0,4459
		10000000	$O(1)$	0,3553
		100000000	$O(1)$	0,3764

Tabla 4. Tiempos de ejecución del método “Desencolar” en Max Heap y cola de prioridad basada en Lista enlazada.

Nombre de la funcionalidad	Nombre de la funcionalidad	Nombre de la funcionalidad	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Encolamiento de valor con menor prioridad a los que estan en la cola (final de la cola)	Max Heap	10000	$O(1)$	0,0154
		100000	$O(1)$	0,0168
		1000000	$O(1)$	0,0169
		10000000	$O(1)$	0,0165
		100000000	$O(1)$	0,0161
	Cola de prioridad basada en Lista enlazada	10000	$O(1)$	0,0102
		100000	$O(1)$	0,0072
		1000000	$O(1)$	0,0082
		10000000	$O(1)$	0,0079
		100000000	$O(1)$	0,0069

Tabla 6. Tiempos de ejecución del método “Encolamiento de valor con menor prioridad a los que están en la cola” en Max Heap y cola de prioridad basada en Lista enlazada.

Nombre de la funcionalidad	Nombre de la funcionalidad	Nombre de la funcionalidad	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Encolamiento de valor con mayor prioridad a los que estan en la cola (inicio de la cola)	Max Heap	10000	$O(\log n)$	0,012
		100000	$O(\log n)$	0,014
		1000000	$O(\log n)$	0,015
		10000000	$O(\log n)$	0,018
		100000000	$O(\log n)$	0,02
	Cola de prioridad basada en Lista enlazada	10000	$O(1)$	0,0085
		100000	$O(1)$	0,0106
		1000000	$O(1)$	0,0111
		10000000	$O(1)$	0,0065
		100000000	$O(1)$	0,0086

Tabla 5. Tiempos de ejecución del método “Encolamiento de valor con mayor prioridad a los que están en la cola” en Max Heap y cola de prioridad basada en Lista enlazada.

Nombre de la funcionalidad	Nombre de la funcionalidad	Nombre de la funcionalidad	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Encolamiento promedio (posiciones internas de la cola)	Max Heap	10000	$O(\log n)$	0,0066
		100000	$O(\log n)$	0,0097
		1000000	$O(\log n)$	0,0127
		10000000	$O(\log n)$	0,0148
		100000000	$O(\log n)$	0,0210
	Cola de prioridad basada en Lista enlazada	10000	$O(n)$	1,7673
		100000	$O(n)$	60,5152
		1000000	$O(n)$	169,1087
		10000000	$O(n)$	655,8323
		100000000	$O(n)$	3.314,6218

Tabla 7. Tiempos de ejecución del método “Encolamiento promedio” en Max Heap y cola de prioridad basada en Lista enlazada.

Análisis comparativo de las Gráficas con respecto a la Complejidad Temporal

	Desencolar	Encolamiento de valor con mayor prioridad a los que están en la cola (inicio de la cola)	Encolamiento de valor con menor prioridad a los que están en la cola (final de la cola)	Encolamiento promedio (posiciones internas de la cola)
Max-Heap	$O(\log n)$	$O(\log n)$	$O(1)$	$O(\log n)$
Colas de prioridad basadas en linkedList	$O(1)$	$O(1)$	$O(1)$	$O(n)$

Puede parecer que la cola basada en lista enlazada es mejor que el montículo pero teniendo en cuenta que la mayoría de veces se encolan datos en las posiciones internas de una cola, la mejor opción para la cola de prioridad es el montículo ya que la operación que más se usa es $O(\log n)$ mientras que la misma operación es $O(n)$ en la cola basada en lista enlazada.

Pruebas y análisis comparativo del uso de las nuevas estructuras de datos implementadas- Tablas Hash.

Comparación de la eliminación de datos en tabla hash de Direccionamiento cerrado y Direccionamiento abierto

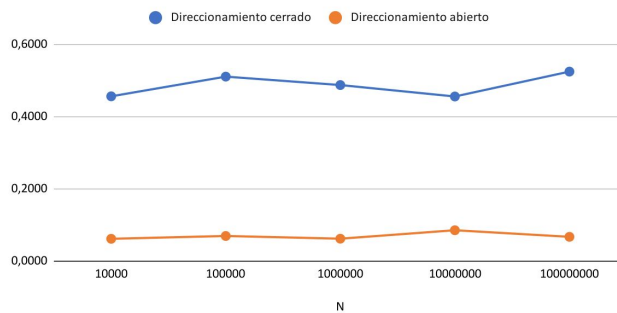


Gráfico 15 . Tiempos de ejecución método “eliminar” en Tablas Hash.

Comparación de la búsqueda de datos en tabla hash de Direccionamiento abierto y Direccionamiento cerrado

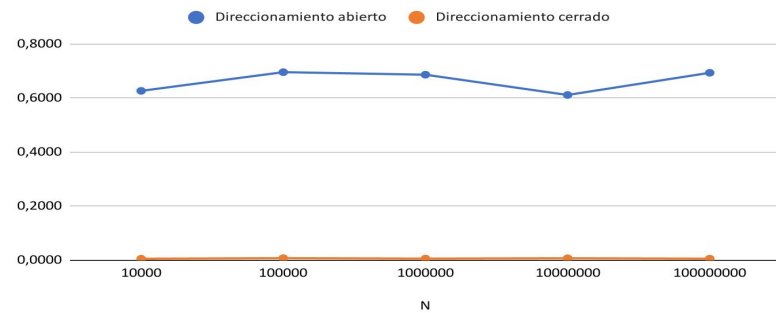


Gráfico 16. Tiempos de ejecución método “búsqueda” en tablas Hash.

Comparación de la inserción de datos en tabla hash de Direccionamiento cerrado y Direccionamiento abierto

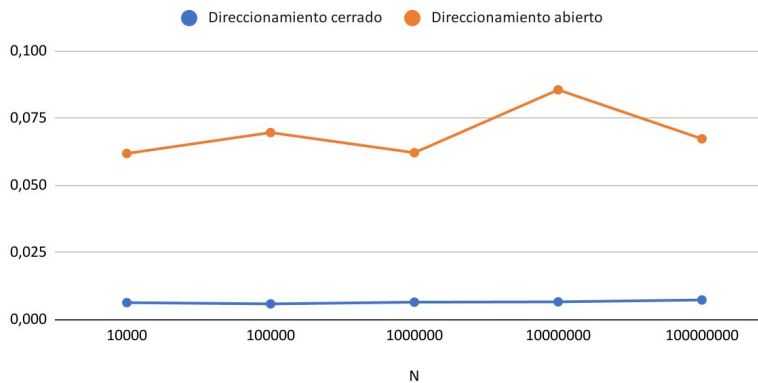


Gráfico 17. Tiempos de ejecución método “inserción” en tablas Hash.

De las Gráficas 15,16 y 17, se concluye que en las tablas hash de direccionamiento abierto y cerrado tienen un crecimiento constante al usar el método “eliminar”, “buscar” e “insertar”.

Pruebas del uso de las nuevas estructuras de datos implementadas

Al igual que se cambio la lista enlazada por el árbol AVL para almacenar la información de las asignaturas, al analizar la tabla hash se decidió cambiar el arbol AVL por la tabla hash para esta funcionalidad.

Nombre de la funcionalidad	Tipo(s) de estructura de datos	Cantidad de datos probados	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Eliminación de datos	Tabla Hash	10000	O(1)	0,4561
		100000	O(1)	0,5106
	Direccional miento cerrado	1000000	O(1)	0,4873
		10000000	O(1)	0,4556
		100000000	O(1)	0,5245
	Tabla Hash	10000	O(1)	0,0044
		100000	O(1)	0,0060
		1000000	O(1)	0,0063
		10000000	O(1)	0,0063
		100000000	O(1)	0,0047

Tabla 8. Tiempos de ejecución del método “Eliminar” en las tablas hash.

Nombre de la funcionalidad	Nombre de la funcionalidad	Nombre de la funcionalidad	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Búsqueda de datos	Tabla Hash	10000	O(1)	0,6268
		100000	O(1)	0,6959
	Direccional miento cerrado	1000000	O(1)	0,6865
		10000000	O(1)	0,6116
		100000000	O(1)	0,6936
	Tabla Hash	10000	O(1)	0,0060
		100000	O(1)	0,0081
		1000000	O(1)	0,0064
		10000000	O(1)	0,0078
		100000000	O(1)	0,0063

Tabla 9. Tiempos de ejecución del método “Buscar” en tablas hash.

Nombre de la funcionalidad	Nombre de la funcionalidad	Nombre de la funcionalidad	Análisis realizado (Notación Big O)	Tiempos de ejecución (ms)
Inserción de datos	Tabla Hash	10000	O(1)	0,0064
		100000	O(1)	0,0059
	Direccional miento cerrado	1000000	O(1)	0,0066
		10000000	O(1)	0,0067
		100000000	O(1)	0,0074
	Tabla Hash	10000	O(1)	0,0619
		100000	O(1)	0,0697
		1000000	O(1)	0,0622
		10000000	O(1)	0,0856
		100000000	O(1)	0,0674

Tabla 10. Tiempos de ejecución del método “insertar” en las tablas Hash.

Análisis comparativo de las Gráficas con respecto a la Complejidad Temporal

Tablas Hash	Complejidad Temporal		
	Eliminar Datos	Buscar Datos	Insertar datos
Close Hash	$O(1)$	$O(1)$	$O(1)$
Open Hash	$O(1)$	$O(1)$	$O(1)$

Ambas implementaciones de la tabla Hash son bastante buenas ya que cuentan con operaciones que ocurren en $O(1)$ pero consideramos mejor al direccionamiento abierto ya que no necesita de una estructura de datos adicional además de que si no se tiene un control adecuado, las listas enlazadas del direccionamiento abierto pueden volverse muy grandes para grandes cantidades de datos ocasionando que las operaciones ya no ocurran en tiempo constante.



Conclusiones

Los tiempos de ejecución de un árbol balanceado(AVL) son notoriamente menores que los del árbol desbalanceado.

La tabla Hash con direccionamiento abierto es superior en las funciones de búsqueda y eliminación de datos con respecto al direccionamiento cerrado, así mismo, la función de inserción es superior en el direccionamiento cerrado, a pesar de que en las dos tablas el tiempo de ejecución es constante para las tres funciones.

La tabla Hash con direccionamiento cerrado, al depender de una estructura de datos adicional (lista enlazada), provoca que algunas funcionalidades sean más lentas con respecto al direccionamiento abierto.

Los tiempos de ejecución en los árboles AVL son menores que los tiempos de ejecución en las listas enlazadas para la mayoría de funciones.



Dificultades y lecciones aprendidas

Definir qué estructuras de datos usar para cada funcionalidad, y con esto aprendimos que es buena práctica hacer varias comparaciones entre diferentes estructuras para definir la “mejor” para cada problema.

Con respecto a las tablas Hash, hubo dificultades, puesto que en ciertas ocasiones calculaba el mismo hash para algunos (diferentes) valores y fue necesario realizar varios ajustes.

Una de las mayores dificultades, fue el tiempo, ya que este estuvo muy limitado, causando que no se pudiesen realizar algunas cosas como una interfaz gráfica para el prototipo.

Una de las lecciones más importantes, fue el aprender a manejar mejor el tiempo, y a distribuirlo de mejor manera para satisfacer todos los requerimientos.