



UNIVERSITAT DE
BARCELONA

Grau d'Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica



Algorísmica

Algorismes de cerca

Mireia Ribera, Jordi Vitrià

Algorismes de cerca

El concepte de cerca inclou diferents conceptes:

- Cerca d'un determinat element en una llista (màx, $x = "a"$, el que compleix una certa condició, etc.).
- Cerca d'un determinat element en una llista ordenada.
- Cerca en un arbre.
- Cerca en un graf.
- Satisfacció de restriccions.
- Etc.

Ens centrarem en la cerca en llistes.

Algorismes de cerca: **cerca lineal**

L'algorisme que implementa amb una estratègia de força bruta la cerca d'un element en una llista es diu **cerca seqüencial o lineal**.

```
def linsearch(list,ele):  
    i=0  
    while i<len(list) and list[i] != ele:  
        i += 1  
    if i<len(list): return i  
    else: return -1
```

La complexitat de l'algorisme és $O(n)$ en el pitjor cas!

Algorismes de cerca: **cerca lineal**

Podem fer una petita millora si afegim l'element que busquem al final de la llista:

```
def linsearch(list,ele):  
    list.append(ele)  
    i=0  
    while list[i] != ele:  
        i += 1  
    if i<len(list)-1: return i  
    else: return -1
```

Algorismes de cerca: **cerca binaria**

I si la llista està ordenada (un diccionari, els nombres de la loteria, etc.) ho podem fer millor?

Lotería Nacional
SORTEO EXTRAORDINARIO DE NAVIDAD
LISTA DE LOS NÚMEROS PREMIADOS EN CADA UNA DE LAS CUENTA OCHENTA SERIES CORRESPONDIENTES AL SORTEO CELEBRADO EN MADRID EL DÍA 22 DE DICIEMBRE DE 2017

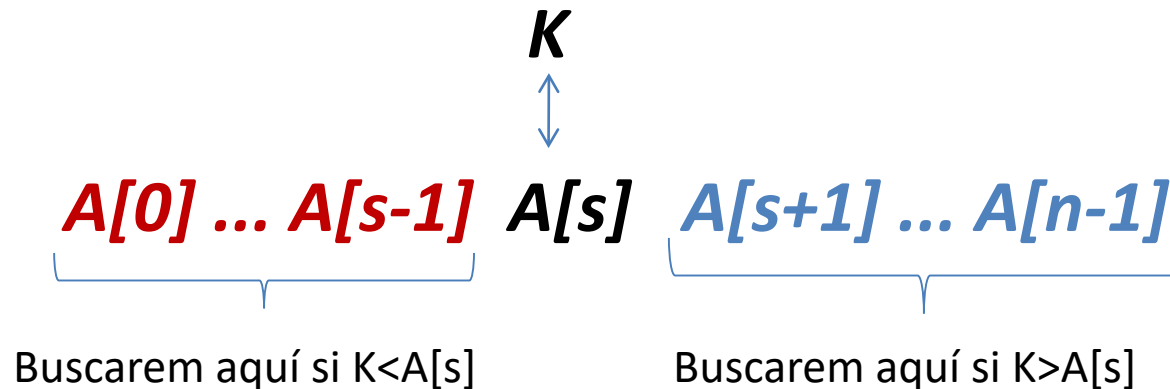
SORTEO NÚM. **102**
DEL AÑO 2017

000000	000001	000002	000003	000004	000005	000006	000007	000008	000009	000010	000011	000012	000013	000014	000015	000016	000017	000018	000019	000020	000021	000022	000023	000024	000025	000026	000027	000028	000029	000030	000031	000032	000033	000034	000035	000036	000037	000038	000039	000040	000041	000042	000043	000044	000045	000046	000047	000048	000049	000050	000051	000052	000053	000054	000055	000056	000057	000058	000059	000060	000061	000062	000063	000064	000065	000066	000067	000068	000069	000070	000071	000072	000073	000074	000075	000076	000077	000078	000079	000080	000081	000082	000083	000084	000085	000086	000087	000088	000089	000090	000091	000092	000093	000094	000095	000096	000097	000098	000099	000100	000101	000102	000103	000104	000105	000106	000107	000108	000109	000110	000111	000112	000113	000114	000115	000116	000117	000118	000119	000120	000121	000122	000123	000124	000125	000126	000127	000128	000129	000130	000131	000132	000133	000134	000135	000136	000137	000138	000139	000140	000141	000142	000143	000144	000145	000146	000147	000148	000149	000150	000151	000152	000153	000154	000155	000156	000157	000158	000159	000160	000161	000162	000163	000164	000165	000166	000167	000168	000169	000170	000171	000172	000173	000174	000175	000176	000177	000178	000179	000180	000181	000182	000183	000184	000185	000186	000187	000188	000189	000190	000191	000192	000193	000194	000195	000196	000197	000198	000199	000200	000201	000202	000203	000204	000205	000206	000207	000208	000209	000210	000211	000212	000213	000214	000215	000216	000217	000218	000219	000220	000221	000222	000223	000224	000225	000226	000227	000228	000229	000230	000231	000232	000233	000234	000235	000236	000237	000238	000239	000240	000241	000242	000243	000244	000245	000246	000247	000248	000249	000250	000251	000252	000253	000254	000255	000256	000257	000258	000259	000260	000261	000262	000263	000264	000265	000266	000267	000268	000269	000270	000271	000272	000273	000274	000275	000276	000277	000278	000279	000280	000281	000282	000283	000284	000285	000286	000287	000288	000289	000290	000291	000292	000293	000294	000295	000296	000297	000298	000299	000300	000301	000302	000303	000304	000305	000306	000307	000308	000309	000310	000311	000312	000313	000314	000315	000316	000317	000318	000319	000320	000321	000322	000323	000324	000325	000326	000327	000328	000329	000330	000331	000332	000333	000334	000335	000336	000337	000338	000339	000340	000341	000342	000343	000344	000345	000346	000347	000348	000349	000350	000351	000352	000353	000354	000355	000356	000357	000358	000359	000360	000361	000362	000363	000364	000365	000366	000367	000368	000369	000370	000371	000372	000373	000374	000375	000376	000377	000378	000379	000380	000381	000382	000383	000384	000385	000386	000387	000388	000389	000390	000391	000392	000393	000394	000395	000396	000397	000398	000399	000400	000401	000402	000403	000404	000405	000406	000407	000408	000409	000410	000411	000412	000413	000414	000415	000416	000417	000418	000419	000420	000421	000422	000423	000424	000425	000426	000427	000428	000429	000430	000431	000432	000433	000434	000435	000436	000437	000438	000439	000440	000441	000442	000443	000444	000445	000446	000447	000448	000449	000450	000451	000452	000453	000454	000455	000456	000457	000458	000459	000460	000461	000462	000463	000464	000465	000466	000467	000468	000469	000470	000471	000472	000473	000474	000475	000476	000477	000478	000479	000480	000481	000482	000483	000484	000485	000486	000487	000488	000489	000490	000491	000492	000493	000494	000495	000496	000497	000498	000499	000500	000501	000502	000503	000504	000505	000506	000507	000508	000509	000510	000511	000512	000513	000514	000515	000516	000517	000518	000519	000520	000521	000522	000523	000524	000525	000526	000527	000528	000529	000530	000531	000532	000533	000534	000535	000536	000537	000538	000539	000540	000541	000542	000543	000544	000545	000546	000547	000548	000549	000550	000551	000552	000553	000554	000555	000556	000557	000558	000559	000560	000561	000562	000563	000564	000565	000566	000567	000568	000569	000570	000571	000572	000573	000574	000575	000576	000577	000578	000579	000580	000581	000582	000583	000584	000585	000586	000587	000588	000589	000590	000591	000592	000593	000594	000595	000596	000597	000598	000599	000600	000601	000602	000603	000604	000605	000606	000607	000608	000609	000610	000611	000612	000613	000614	000615	000616	000617	000618	000619	000620	000621	000622	000623	000624	000625	000626	000627	000628	000629	000630	000631	000632	000633	000634	000635	000636	000637	000638	000639	000640	000641	000642	000643	000644	000645	000646	000647	000648	000649	000650	000651	000652	000653	000654	000655	000656	000657	000658	000659	000660	000661	000662	000663	000664	000665	000666	000667	000668	000669	000670	000671	000672	000673	000674	000675	000676	000677	000678	000679	000680	000681	000682	000683	000684	000685	000686	000687	000688	000689	000690	000691	000692	000693	000694	000695	000696	000697	000698	000699	000700	000701	000702	000703	000704	000705	000706	000707	000708	000709	000710	000711	000712	000713	000714	000715	000716	000717	000718	000719	000720	000721	000722	000723	000724	000725	000726	000727	000728	000729	000730	000731	000732	000733	000734	000735	000736	000737	000738	000739	000740	000741	000742	000743	000744	000745	000746	000747	000748	000749	000750	000751	000752	000753	000754	000755	000756	000757	000758	000759	000760	000761	000762	000763	000764	000765	000766	000767	000768	000769	000770	000771	000772	000773	000774	000775	000776	000777	000778	000779	000780	000781	000782	000783	000784	000785	000786	000787	000788	000789	000790	000791	000792	000793	000794	000795	000796	000797	000798	000799	000800	000801	000802	000803	000804	000805	000806	000807	000808	000809	000810	000811	000812	000813	000814	000815	000816	000817	000818	000819	000820	000821	000822	000823	000824	000825	000826	000827	000828	000829	000830	000831	000832	000833	000834	000835	000836	000837	000838	000839	000840	000841	000842	000843	000844	000845	000846	000847	000848	000849	000850	000851	000852	000853	000854	000855	000856	000857	000858	000859	000860	000861	000862	000863	000864	000865	000866	000867	000868	000869	000870	000871	000872	000873	000874	000875	000876	000877	000878	000879	000880	000881	000882	000883	000884	000885	000886	000887	000888	000889	000890	000891	000892	000893	000894	000895	000896	000897	000898	000899	000900	000901	000902	000903	000904	000905	000906	000907	000908	000909	000910	000911	000912	000913	000914	000915	000916	000917	000918	000919	000920	000921	000922	000923	000924	000925	000926	000927	000928	000929	000930	000931	000932	000933	000934	000935	000936	000937	000938	000939	000940	000941	000942	000943	000944	000945	000946	000947	000948	000949	000950	000951	000952	000953	000954	000955	000956	000957	000958	000959	000960	000961	000962	000963	000964	000965	000966	000967	000968	000969	000970	000971	000972	000973	000974	000975	000976	000977	000978	000979	000980	000981	000982	000983	000984	000985	000986	000987	000988	000989	000990	000991	000992	000993	000994	000995	000996	000997	000998	000999	001000	001001	001002	001003	001004	001005	001006	001007	001008	001009	001010	001011	001012	001013	001014	001015	001016	001017	001018	001019	001020	001021	001022	001023	001024	001025	001026	001027	001028	001029	001030	001031	001032	001033	001034	001035	001036	001037	001038	001039	001040	001041	001042	001043	001044	001045	001046	001047	001048	001049	001050	001051	001052	001053	001054	001055	001056	001057	001058	001059	001060	001061	001062	001063	001064	001065	001066	001067	001068	001069	001070	001071	001072	001073	001074	001075	001076	001077	001078	001079	001080	001081	001082	001083	001084	001085	001086	001087	001088	001089	001090	001091	001092	001093	001094	001095	001096	001097	001098	001099	001100	001101	0011
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	------

Algorismes de cerca: **cerca binaria**



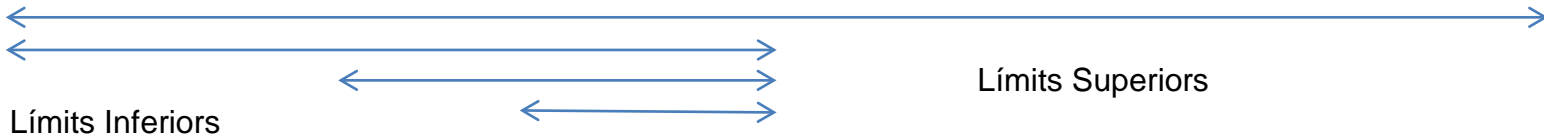
La **cerca binaria** ho fa comparant l'element cercat K a l'element central de la llista: si hi ha correspondència ja l'hem trobat, sinó, busquem a la subllista que correspon.



Find "J"

Valor Mig

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X



Com acaba l'algorisme si J no hi és?

Algorismes de cerca: **cerca binaria**

```
def recbinsearch(k, nums, low, high):  
  
    # low, high defineixen els limits de la llista  
    # on buscar, així no cal crear noves llistes  
  
    if low > high: return -1  
    mid = (low + high) // 2  
    items = nums[mid]  
    if items == k: return mid  
    elif k < items:  
        return recbinsearch(k, nums, low, mid-1)  
    else: return recbinsearch(k, nums, mid+1, high)  
  
recbinsearch(3, [1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 8)  
  
>>> 2
```


Algorismes de cerca: **cerca binaria**

Anem a veure com funcionaria per $K=70$.

Índex	0	1	2	3	4	5	6	7	8	9	10	11	12
Valor	3	14	27	31	39	42	55	70	74	81	85	93	98
It 1	<i>l</i>						<i>m</i>						<i>h</i>
It 2								<i>l</i>		<i>m</i>			<i>h</i>
It 3								<i>l,m</i>	<i>h</i>				

Algorismes de cerca: **cerca binaria**

Per analitzar la seva complexitat calcularem el nombre de vegades que la **clau de la cerca, K** , es compara amb un element de la llista.

En el **pitjor dels casos** (quan l'element no hi és), tenim aquesta relació de recurrència:

$$C_{pitjor} = 1 \left(\frac{n}{2} \right) + 1$$

Segons el teorema Màster això és $O(\log_2 n)$: per una llista de 1.000.000 elements són 20 comparacions!

Algorismes de cerca: **cerca binaria**

Evidentment és un algorisme **recursiu**, però és pot implementar fàcilment de forma **no recursiva**.

```
def binsearch(nums, K):  
    low = 0  
    high = len(nums)-1  
    while low <= high:  
        mid = (low + high) // 2  
        if nums[mid] > K: high = mid - 1  
        elif nums[mid] < K: low = mid + 1  
        else: return mid  
    return -1
```

Algorismes de cerca: **cerca binaria**

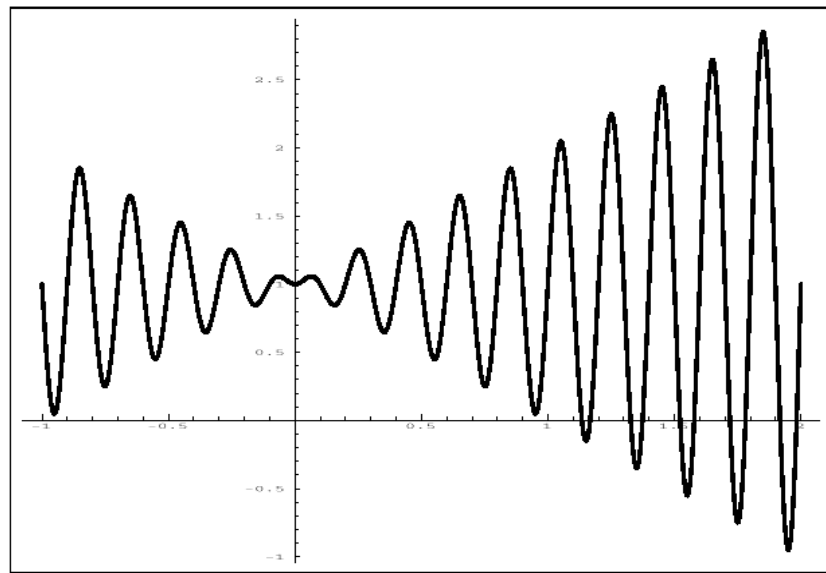
El **cas promig** és més difícil d'analitzar, però es pot demostrar que és només una mica millor que el pitjor cas (tot i que del mateix ordre).

Observació: Si tenim una llista desordenada de mida n i només hem de buscar un element (o pocs), apliquem una cerca exhaustiva. Però si hem de fer moltes cerques (de l'ordre de n), val la pena ordenar-la primer i fer cerca binària dels elements després!

Altres algorismes de cerca (funcions)

Imaginem ara que tenim un **vector no ordenat**, com pot ser el corresponent als valors discrets d'una **funció multimodal**.

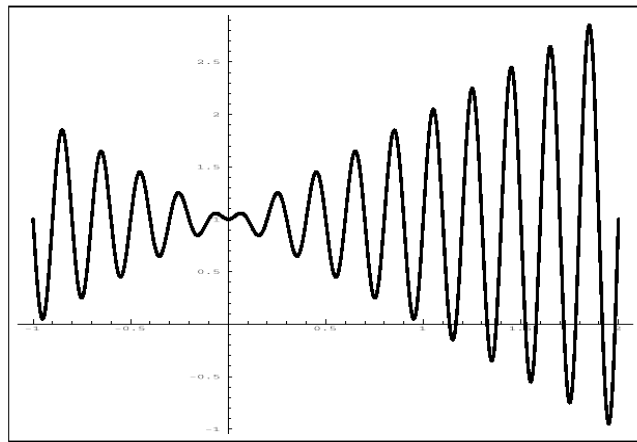
Com busquem el màxim? Quin tipus de cerca hi podem aplicar?



Algorismes de cerca

Podem aplicar-hi **cerca exhaustiva**:

```
def func1d(x): #funció multimodal de la que buscarem max
    import math
    y = x * math.sin(10*math.pi*(x))+1.0
    return y
```



Algorismes de cerca

La funció range només genera enters

```
def frange(start, stop, step):  
    current = start  
    while current < stop:  
        yield current  
        current += step
```

```
def linsearchfunc1d():  
    x=0.0      #punt x que dóna valor max  
    maxim=0.0  #func1d(x)  
    for i in frange(-1.0,2.0,0.01):  
        if func1d(i)>maxim:  
            maxim=func1d(i)  
            x=i  
    print(maxim)
```

En aquest cas analitzem
300 punts

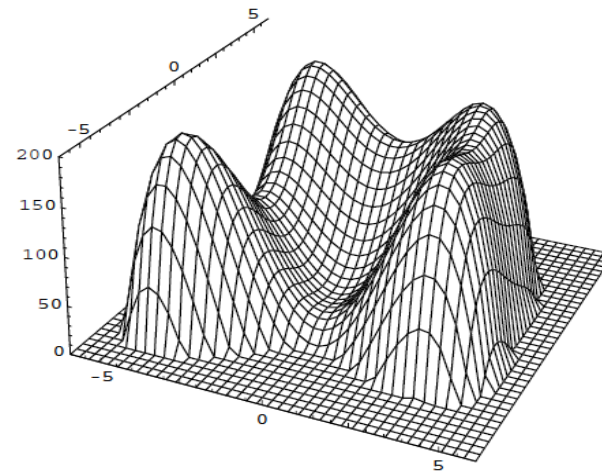
La complexitat és $O(n)$ on
 n depèn de la precisió
que volem

Algorismes de cerca: **cerca aleatòria**

Si el nombre de punts a mostrejar és molt gran tenim un problema!

En aquest cas, amb precisió 0.01 analitzaríem 300x300 punts

Si necessitéssim una precisió 0.000001 analitzaríem 3000000x3000000 punts

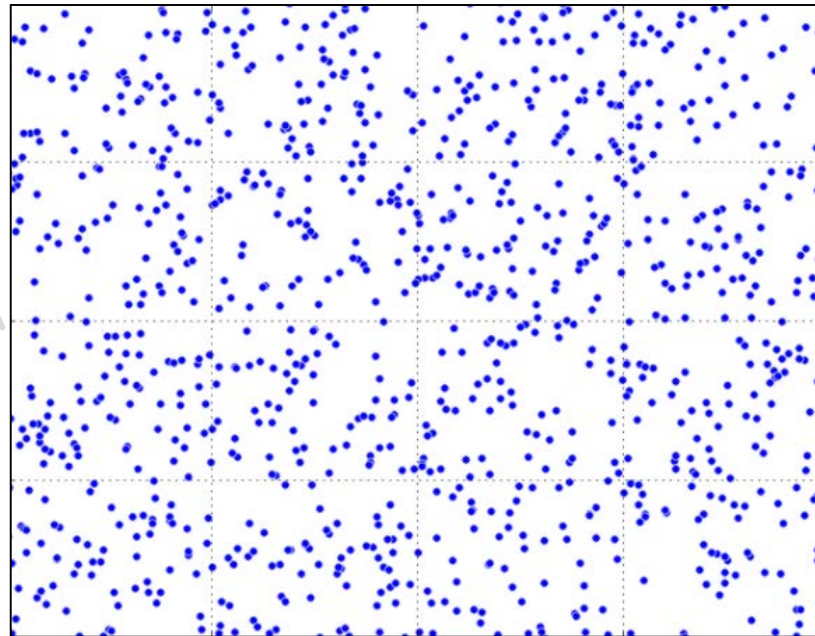


No podem fer cerca exhaustiva!

Algorismes de cerca: **cerca aleatòria**

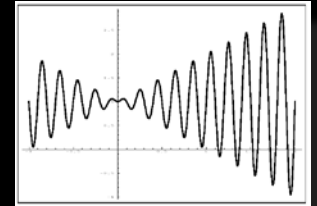
Té sentit fer una **cerca aleatòria**? (= anar generant nombres de forma aleatòria dins del rang de les variables i quedar-se el màxim).

Mostra aleatòria
sobre el pla.



Algorismes de cerca: **cerca aleatòria**

```
def rsearchfunc1d():  
    import random  
    x=0.0  
    maxim=0.0  
    for i in range(1000):  
        xtemp = (random.random()*3.0)-1.0  
        if func1d(xtemp)>maxim:  
            maxim=func1d(xtemp)  
            x=xtemp  
    return maxim
```



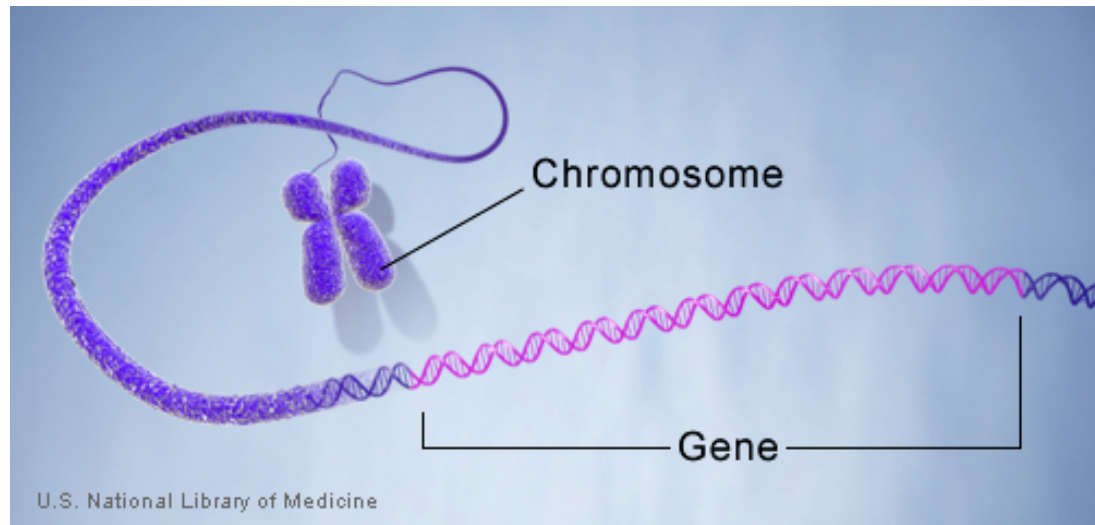
100 proves	2.77824636954	1.000 proves	2.85027049997	100.000 proves	2.8502736913
	2.76633333502		2.84068071726		2.85026970833
	2.49830837751		2.82585079483		2.85026429332
	2.84180575738		2.82441879719		2.8502737353
	2.67472858999		2.84078409458		2.8502710006
	2.84721009237		2.83748038425		2.85026302851
	2.60189299072		2.84883426411		2.85027375351
	2.81619415008		2.8497277592		2.85023546116
	2.81493367995		2.84184730168		2.85027214716
	2.64975396079		2.84990510016		2.8502538051

Algorismes de cerca: **cerca aleatòria**

En general, la cerca totalment aleatòria **no és una bona solució**: tenim el cost de la cerca afitat, però depèn molt de l'aleatorietat i té un **resultat molt semblant, si no equivalent, a la cerca lineal, per força bruta.**

Algorismes de cerca: **cerca amb algorismes genètics.**

Anem a veure un tipus **d'algorisme aproximat** que ens fa una cerca, amb **un cert component aleatori** més intel·ligent, de l'espai de solucions: la cerca basada en **algorismes genètics.**



Algorismes genètics

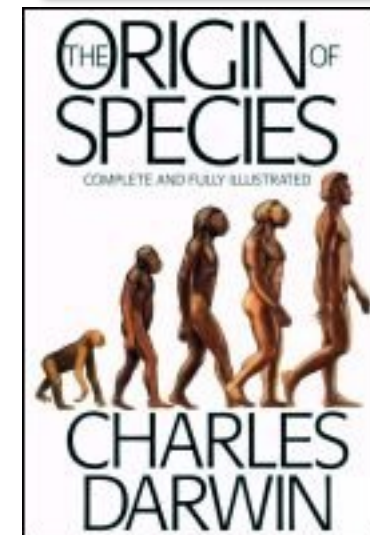
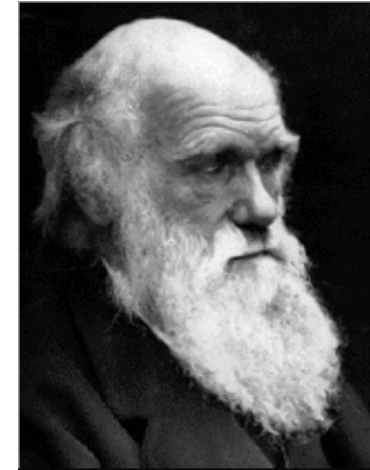
Algorismes de cerca: **cerca amb algorismes genètics.**

Precedents

El terme **algorismes genètics** s'utilitza per a referir-se a una família bastant àmplia de models computacionals de càlcul basats en els mecanismes d'evolució biològica.

La idea de **selecció natural** va ser introduïda per Charles Darwin el 1859 dins del seu llibre "L'Origen de Les Espècies".

Aquesta idea pot servir d'analogia per a construir mètodes de **cerca** en problemes d'optimització combinatòria i mètodes d'**aprenentatge**.



Algorismes de cerca: **cerca amb algorismes genètics.**

Precedents

Darwin va assentar les bases del principi d'evolució per selecció natural amb les següents idees:



- Cada individu tendeix a passar els seus trets característics a la seva descendència.
- Tot i així, la natura produeix individus amb trets diferents.
- Els individus més adaptats tendeixen a tenir més descendència, i a la llarga, la població tendeix a ser "millor".
- Sobre grans períodes, l'acumulació de canvis pot produir espècies totalment noves, adaptades al seu entorn.

A més a més la natura disposa d'una sèrie de mecanismes reguladors externs a aquest procés però igualment interessants: el mecanisme de diversitat, els paràsits, les organitzacions socials, etc.

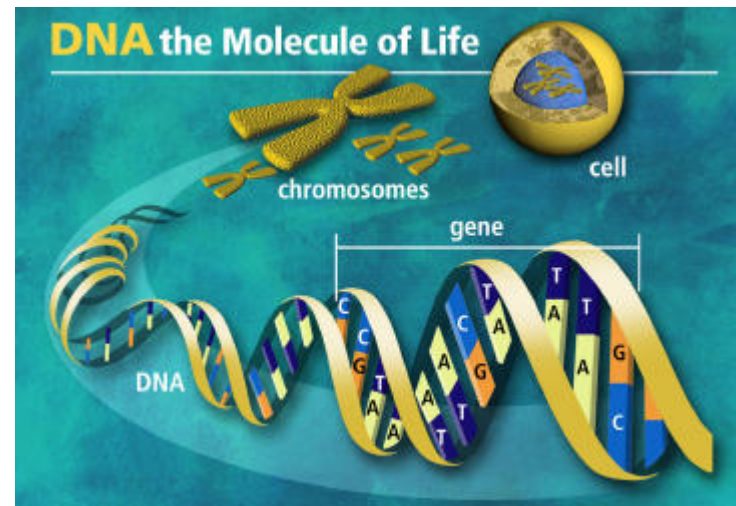
Algorismes de cerca: **cerca amb algorismes genètics.**

Precedents

Els mecanismes biològics que fan possible l'evolució són avui coneguts. A la natura, podem veure com la transmissió de la informació genètica (genoma) es fa a través de la reproducció sexual. Aquest procediment permet als descendents ser diferents dels seus antecessors, tot i que conservant la majoria de trets. El mecanisme sobre el que està basat això es troba a nivell molecular, i consisteix en l'aparellament de cromosomes (lloc on trobem el genoma), l'intercanvi d'informació, i la posterior partició. D'això n'hi direm **creuament**. La probabilitat de que dos individus es creuin depèn de la seva **adaptació** al medi.

Per inspiració d'aquests mecanismes usarem terminologia de biologia per als nostre problemes:

- Gens
- Genoma
- Cromosomes
- Creuaments i mutacions.
- Funció d'adaptació.
- Mecanismes correctors/moduladors: diversitat, parasitisme, organització social, etc.



Algorismes de cerca: **cerca amb algorismes genètics.**

El **cicle normal** d'un algorisme genètic és:

- **avaluar** l'adaptació de tots els individus de la població (amb la **funció d'adaptació**). Aquesta funció incorpora l'objectiu del problema.
- **crear** una nova població mitjançant reproducció fent servir:
 - **creuament** +
 - **mutació** dels cromosomes dels individus + descartar la població antiga (hem d'assegurar que el resultat d'aplicar els operadors genera **possibles solucions al problema**.)
- **iterar** sobre la nova població.

Cada una de les iteracions d'aquest cicle es coneix com **generació**.

Algorismes de cerca: **cerca amb algorismes genètics.**

Disseny d'algorismes genètics

Caldrà decidir algunes qüestions:

1. Quina és la **funció d'adaptació**?
2. Com **representarem** els individus/solucions?
3. Com **seleccionarem** els individus per **reproduir-se**?
4. Com **creuarem** i **mutarem** els individus?
5. Quina és la **probabilitat de mutació**?
6. Necessitem **mecanismes moduladors** (p.e. diversitat)?

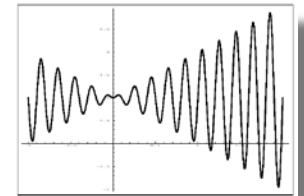
Algorismes de cerca: **cerca amb algorismes genètics.**

Funció d'adaptació

La funció d'adaptació és **pròpia de cada problema** que volem resoldre.

En el problema que hem posat com a exemple, la funció d'adaptació és el valor de $f(x)$:

$$(\text{math.sin}(10*\text{math.pi}*(x))+1.0)$$



Per tant, la màxima adaptació correspon al màxim d'aquesta funció multimodal

El problema del viatjant de comerç és un problema candidat a ser resolt amb algorismes genètics.

La funció d'adaptació seria $1/d$, on d és la distància recorreguda (i així un valor de la funció alt és una bona solució).

Un cromosoma representaria un circuit que és potencialment solució del problema.

Algorismes de cerca: **cerca amb algorismes genètics.**

Representació

Normalment es considera que la millor representació dels cromosomes possible és la **binària.**

El creuament, la mutació, i d'altres operacions que es poden utilitzar, són aleshores simples **operacions a nivell de bit.**

Algorismes de cerca: **cerca amb algorismes genètics.**

Suposem doncs que tenim una població inicial de quatre individus amb les següents característiques:

Individu	Valor Adaptació	Probabilitat de Selecció
0 0 0 1 1 0 0 1 0 1 1 1	8	32%
1 1 1 0 1 0 1 0 1 1 0 0	6	24%
0 0 1 1 1 0 1 0 1 0 0 1	6	24%
1 1 1 0 1 1 0 1 1 1 0 0	5	20%

I suposem que definim la **probabilitat de selecció** com:

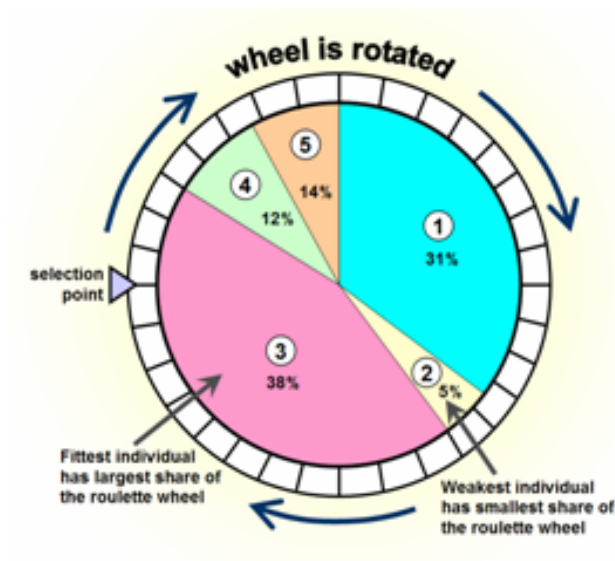
$$f_i = \frac{q_i}{\sum_j q_j}$$

Com els **seleccionem** i els **creuem**?

Algorismes de cerca: **cerca amb algorismes genètics.**

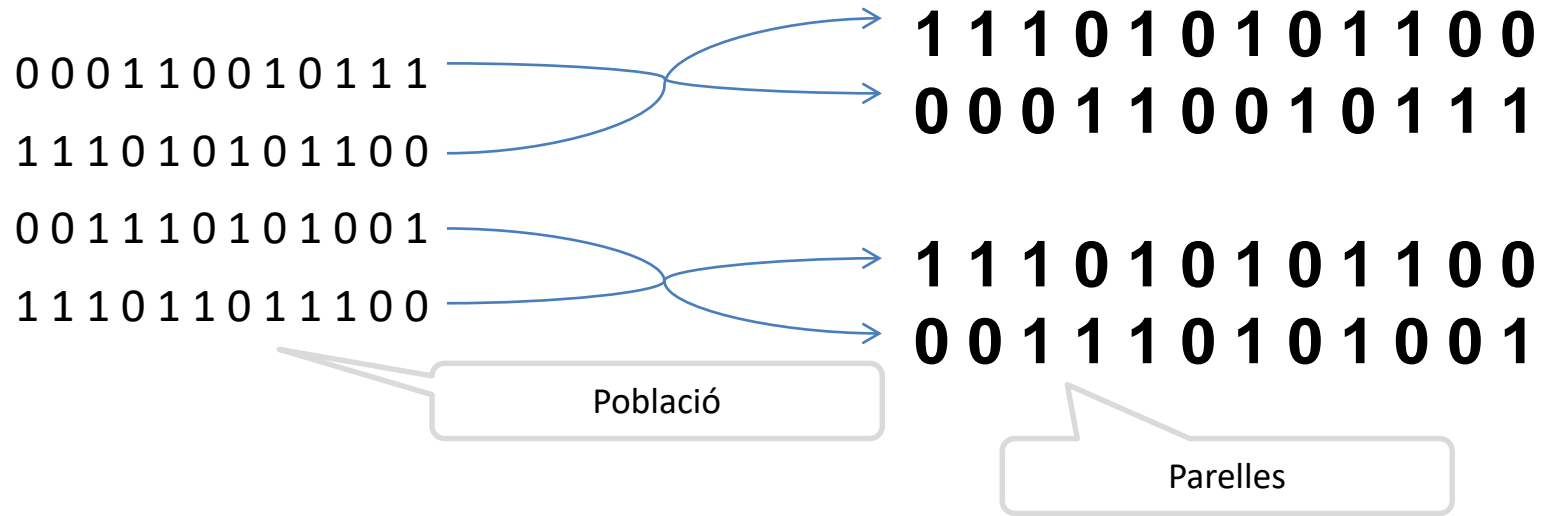
Una primera alternativa per la **selecció** és triar parelles aleatòriament **tenint en compte la seva probabilitat de selecció.**

És com si fem rodar una ruleta i ens va donant individus; l'individu amb més probabilitat de selecció ocupa més posicions a la ruleta.



Algorismes de cerca: **cerca amb algorismes genètics.**

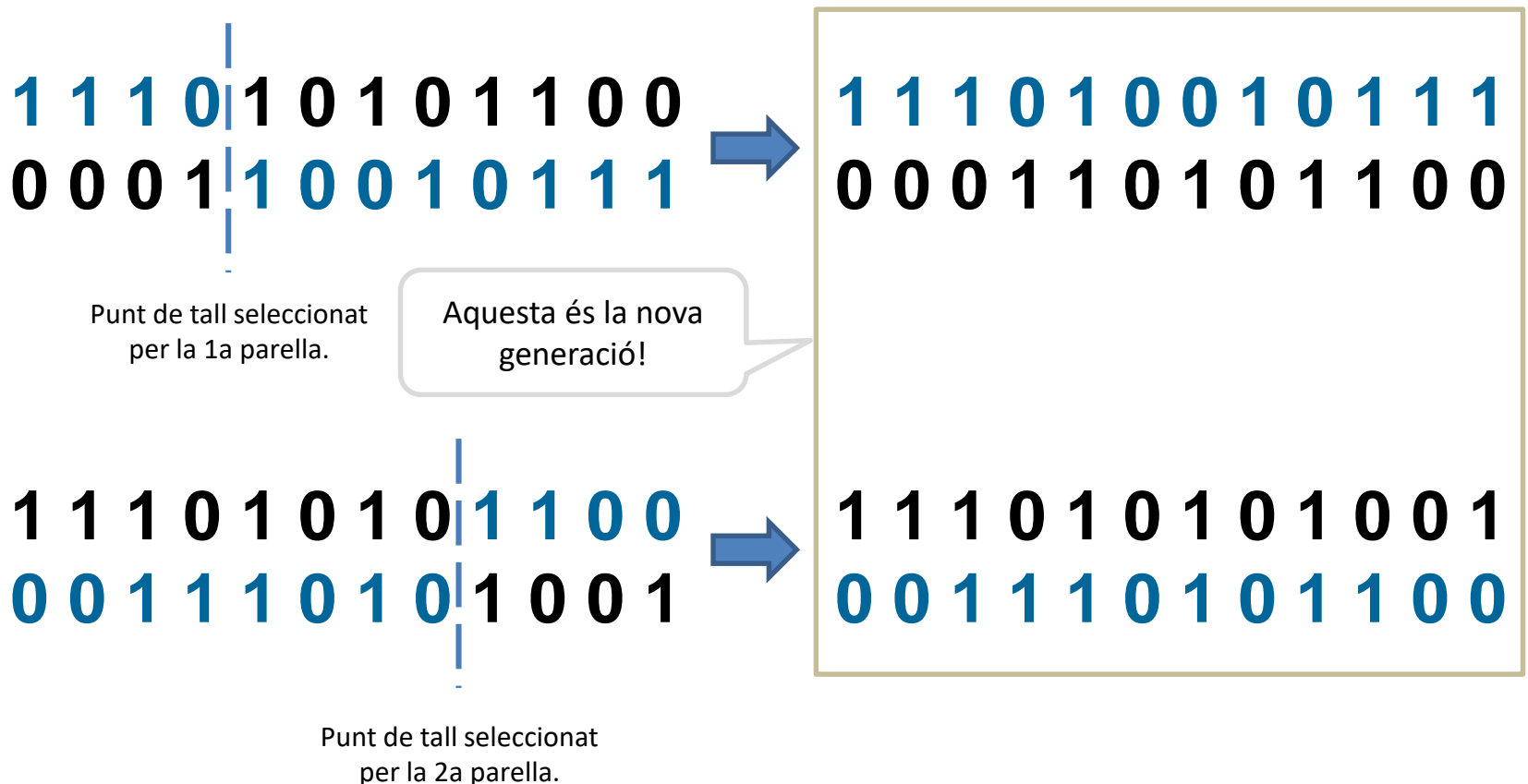
Imaginem que la selecció ens ha donat aquestes dues parelles:



I ara, com les **creuem**?

Algorismes de cerca: **cerca amb algorismes genètics.**

La forma més simple de creuament és generar un punt de tall aleatòriament i intercanviar:



Algorismes de cerca: **cerca amb algorismes genètics.**

Per a **mutar-los** canviarem el valor d'un quants bits de la població de forma aleatòria.

La probabilitat de que un bit canvii de valor és β i la que probabilitat de no canviar és $(1 - \beta)$, però **sempre $\beta \ll (1 - \beta)$**

1	1	1	0	1	0	0	1	0	1	1	1
0	0	0	1	1	0	1	0	1	1	0	0
1	1	1	1	1	0	1	0	1	0	0	1
0	0	1	1	1	0	1	0	1	1	0	1

Algorismes de cerca: **cerca amb algorismes genètics.**

Resum fins ara...

La **funció d'adaptació** depèn del problema que volem resoldre.

La **representació** òptima, és en la majoria de casos i si no hi ha motius fonamentats per dubtar-ho, la **binària**.

La **representació** ha facilitar que el resultat d'aplicar els operadors genètics sigui vàlid.

Les restriccions sobre la representació no formen part de l'algorisme genètic!

L'operació de **creuament** crea dos nous individus seleccionant punts de creuament en els cromosomes seleccionats i intercanviant les seves parts.

L'operació de **mutació** consisteix en la selecció aleatòria d'algun dels gens del cromosoma i el canvi del seu valor. La probabilitat de mutació ha de ser petita (si no ho convertim en cerca aleatòria!).

Algorismes de cerca: **cerca amb algorismes genètics.**

Probabilitat de supervivència

- El **valor d'adaptació** de cada individu depèn del problema concret.
- La **probabilitat de supervivència a la següent generació** és una **ponderació del valor d'adaptació**, i es pot fer de diverses maneres: el **mètode estàndard** (que ja hem vist), el **mètode d'ordenació**, el **mètode de la diversitat**, etc.
- A partir d'ara suposarem que la generació següent es forma **a partir d'una selecció entre els elements del conjunt format pels cromosomes progenitors i pels cromosomes descendents**, però seguint una **estratègia elitista**: el(s) millor(s) cromosomes passen automàticament (així assegurem que una bona solució no es perd mai).

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode Estàndard. Donat un cromosoma i , aquest és avaluat com a possible solució al problema en qüestió. Com a resultat obté un **valor d'adaptació** q_i . Llavors definim la seva **probabilitat de selecció** com:

Individu	Valor Adaptació	Probabilitat de Selecció
0 0 0 1 1 0 0 1 0 1 1 1	8	32%
1 1 1 0 1 0 1 0 1 1 0 0	6	24%
0 0 1 1 1 0 1 0 1 0 0 1	6	24%
1 1 1 0 1 1 0 1 1 1 0 0	5	20%

$$f_i = \frac{q_i}{\sum_j q_j}$$

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode Estàndard, problemes.

Un dels inconvenients associat al mètode estàndard és el **poc pes que dóna al cromosomes "*dolents*",** fet que els impedeix de passar a les futures generacions, i per tant, transmetre les poques coses que tinguin bones.

Un altre possible inconvenient és que moltes vegades la **funció d'avaluació és qualitativa:** ordena de forma correcta però els seus valors no són precisos.

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode d'Ordenació.

Per insensibilitzar el mètode de selecció respecte al valor d'adaptació del problema, podem **ordenar** els cromosomes segons aquest valor, i els ponderem segons aquesta regla:

- *Establim un valor aleatori p entre 0 i 1*
- *Al primer cromosoma li assignem aquesta probabilitat*
- *Al segon li assignem la probabilitat $p \cdot (1-p)$*
- *La probabilitat de l' i -èssim cromosoma és $p \cdot (1 - \text{la probabilitat que s'hagi triat algun cromosoma anterior})$.*

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode d'Ordenació.

$0.333 = 1 - (0.667)$ és la probabilitat de que no hagi sortit el primer cromosoma.

Per exemple, suposem que $p=0.667$. Llavors:

<i>Crom (x,y)</i>	q_i	<i>Ordre</i>	<i>Prob M. Estànd.</i>	<i>Prob M. Ordenació</i>
0001,0100	44	1	0.22	0.667
0011,0001	32	2	0.16	$0.222 = 0.667 \times 0.333$
0001,0010	22,5	3	0.125	$0.073 = 0.667 \times 0.111$
0001,0001	1,5	4	0.075	$0.025 = \dots$
0111,0101	0	5	0.0	0.012

Imaginem que aquests cromosomes representen punts del pla i estem buscant el màxim

$0.111 = 1 - (0.667 + 0.222)$ és la probabilitat de que no hagi sortit ni el primer ni el segon cromosoma.

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode de Diversitat.

Aquest mètode es basa en l'anomenat **principi de diversitat**: és quasi tan bo ser diferent com estar adaptat.

Definim la diversitat d'un grup de cromosomes com:

La distància que usem pot ser des del nombre de bits diferents entre cada cromosoma a una funció definida per l'usuari a partir del coneixement del problema.

$$Div = \sum_i \frac{1}{d_i^2}$$

En el nostre exemple considerarem la distància euclidiana sobre punts del pla. En el cas més general podríem fer servir la distància de Hamming.

on d_i és una **mesura de distància entre cromosomes.**

Algorismes de cerca: **cerca amb algorismes genètics.**

La distància que usem pot ser des del nombre de bits diferents entre cada cromosoma a una funció definida per l'usuari a partir del coneixement del problema.

$$Div = \sum_i \frac{1}{d_i^2}$$

En el nostre exemple considerarem la distància euclidiana sobre punts del pla. En el cas general podríem fer servir la distància de Hamming.

La **distància de Hamming** entre dues cadenes de la mateixa longitud és el nombre de posicions diferents. Si considerem cadenes de bits, correspon al nombre de bits que s'han de canviar d'una cadena perquè passi a tenir el valor d'una altra cadena.

Algorismes de cerca: cerca amb algorismes genètics.

Ponderació del valor d'adaptació.

Mètode de Diversitat.

Com l'apliquem?

1. El millor cromosoma passa automàticament a la següent generació (estratègia elitista).
2. Calculem la diversitat de tots els cromosomes respecte als que han passat a la següent generació.
3. Ordenem els cromosomes segons la seva funció d'avaluació.
4. Sumem els nombres que representen l'ordre obtingut per cada cromosoma als passos 2 i 3. I reordenem segons aquest valor.
5. Triem el cromosoma que passa a la següent generació segons el mètode d'ordenació i si queden cromosomes per triar, i tornem al punt 2.

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode de Diversitat.

Exemple:

<i>Cromosomes</i>	<i>q_i</i>
0100 0001	100
0001 0100	44
0011 0001	32
0001 0010	22,5
0001 0001	1
0111 0101	0

El cromosoma millor passa a la següent generació. En el nostre cas és el cromosoma (0100 0001).

Per tant resten per triar 2 cromosomes entre (0001 0100), (0011 0001), (0001 0010), (0001 0001), (0111 0101).

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode de Diversitat.

Exemple:

Construïrem la taula segons el mètode de diversitat segons la diversitat de cada cromosoma respecte al que ja ha passat:

<i>Cromosomes</i>	<i>diversitat</i>	<i>ord div</i>	<i>ord est</i>	<i>div+est</i>	<i>Probab.</i>
0001 0100	0.040	1	1	2	0.667
0011 0001	0.250	5	2	7	0.073
0001 0010	0.059	3	3	6	0.222
0001 0001	0.062	4	4	8	0.012
0111 0101	0.050	2	5	7	0.025

La distància que usem pot ser des del nombre de bits diferents entre cada cromosoma a una funció definida per l'usuari a partir del coneixement del problema.

En aquest cas, la diversitat és només la inversa de la distància euclidiana 2D de cada cromosoma al que ja ha passat.

Algorismes de cerca: **cerca amb algorismes genètics.**

Ponderació del valor d'adaptació.

Mètode de Diversitat.

Exemple:

Llavors triem aleatòriament el següent que passa, i resulta que és el cromosoma (0001 0100). A partir d'aquest moment repetim el procés anterior, però calculant la diversitat respecte al dos cromosomes que ja han passat:

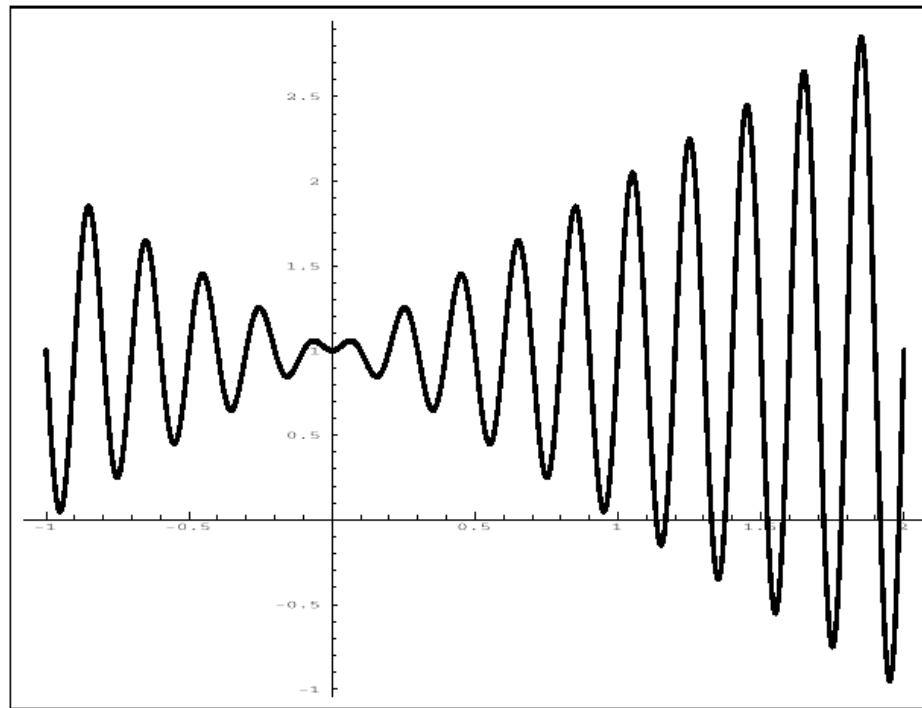
<i>Cromosomes</i>	<i>diversitat</i>	<i>ord div</i>	<i>ord est</i>	<i>div+est</i>	<i>Prob</i>
00110001	0.327	4	1	5	0.667
00010010	0.309	3	2	5	0.222
00010001	0.173	2	3	5	0.073
01110101	0.077	1	4	5	0.025

En el cas d'empats per a l'ordenació resultat de la suma div+ord desempatem segons el valor d'ordenació pura.

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

$$f(x) = x \sin(10\pi x) + 1.0$$



El problema és trobar la x dins del rang $[-1 .. 2]$ que maximitza f .

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

- Utilitzarem un **vector binari** com a cromosoma per a representar el valor real de la variable x . La longitud del vector dependrà del domini i la precisió.
- En el cas estudiat, el domini de la variable x és $[-1,2]$, té longitud 3.
- Suposem que volem 6 decimals (1.000.000 de valors per cada unitat).
- Per tant, necessitem mostrejar el rang en 3.000.000 posicions, o sigui, 22 bits:
 - $2.097.152 = 2^{21} < 3.000.000 < 2^{22} = 4.194.304$

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

La transformació d'una seqüència binària $[b_{21}, \dots, b_0]$ a un nombre real x es fa en dos passos:

1. Primer convertim la seqüència de base 2 a base 10:

$$([b_{21}, \dots, b_0])_2 = \left(\sum_{i=0}^{21} b_i 2^i \right)_{10} = x'$$

2. Després trobem el nombre real corresponent:

$$x = -1.0 + x' \frac{3}{2^{22} - 1}$$

on -1.0 és el límit esquerra de l'interval i 3 la longitud.

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

Escollim com a població inicial 50 individus de forma aleatòria.

```
# Creem la població inicial
def initpop(n, long):
    import random

    # Generem una població de n cromosomes de longitud long.
    pop = [[0] * long for x in range(n)]
    for i in range(n):
        for j in range(long):
            if random.random() > 0.5: pop[i][j] += 1
    return pop
```

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

La funció d'avaluació serà equivalent a la funció del gràfic f :

$v1 = (1000101110110101000111), \text{ cost}(v1) = 1.5886345$

$v2 = (0000001110000000010000), \text{ cost}(v2) = 0.078878$

$v3 = (1110000000111111000101), \text{ cost}(v3) = 2.250650$

```
# Definim la funció d'avaluació d'un cromosoma de len(r) bits.
def cost(r):
    import math
    # Transformem els bits en un valor real a l'interval [-1,2]
    sum=0.0
    for i in range(len(r)): #convertim base 10
        sum = sum + r[i]*(2**i)
    x = -1.0 + sum * (3.0/(2.0**(len(r))-1.0))
        # convertim a flotant entre -1 i 3
        # amb r dígits decimals
    # Avaluem el cromosoma
    y = x * math.sin(10*math.pi*(x))+1.0
    return y
```

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

Per fer la **mutació** imposem una probabilitat de mutació $p_m = 0.01$ per a cada bit.

Per exemple, si tenim el cromosoma

$v_3 = (1110000000111111000101)$

i seleccionem el cinquè bit per mutar, obtindrem

$v_3' = (1110**1**000000111111000101).$

Aquest cromosoma representa el valor $x_3' = 1.721638$, i per tant $f(x_3') = 2.343555$, que s'ha incrementat respecte $f(x_3) = 2.250650$.

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

```
# Definim la mutació amb probabilitat mutprob
def mutacio(r,mutprob):
    import random
    for i in range(len(r)):
        if random.random()<mutprob:
            if r[i]==0: r[i]=1
            else: r[i]=0
    return r
```

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

Per al **creuament** d'una parella de cromosomes escollirem aleatòriament un punt de tall i intercanviarem informació per crear els dos descendents.

```
# Definim el creuament
def creuament(r1,r2):
    import random
    i=random.randint(1,len(r1)-2)
    return r1[:i]+r2[i:],r1[i:]+r2[:i]
```

Algorismes de cerca: **cerca amb algorismes genètics.**

Exemple: Optimització d'una funció multimodal

Si iterem l'algorisme 150 generacions trobem que el millor cromosoma és

$v_{\max} = (1111001101000100000101)$,
que correspon al valor $x_{\max} = 1.850773$.

L'evolució de l'algorisme es pot avaluar a partir del millor valor de la funció aconseguit en cada generació.

Generació	Avaluació de f
1	1.441942
6	2.150003
8	2.250283
9	2.250284
10	2.250363
12	2.328077
39	2.344251
51	2.733893
99	2.849246
137	2.850217
145	2.850227

Algorismes de cerca: cerca amb algorismes genètics.

Exemple: Optimització d'una funció multimodal

Observació vàlida per a qualsevol problema de recerca amb algorismes genètics:

Quin és el factor que més pesa en el càlcul de la complexitat computacional de l'algorisme? El nombre d'avaluacions!

En el nostre problema hem fet
 $50 \times 150 = 7.500$ avaluacions!

Els operadors genètics tenen un cost computacional nul, per tant la complexitat de l'algorisme és el nombre d'avaluacions per la complexitat de l'avaluació.

Annex: Truc de representació binària

En el cas concret de representar en forma binària nombre naturals, s'ha vist que hi ha una codificació que per alguns problemes funciona millor que la clàssica: el **codis de Gray**.

Aquests codis representen cada nombre de la seqüència $0...2^N$ com una seqüència de bits de longitud N tal que dos sencers adjacents tenen una representació que difereix només en un bit. D'això se'n diu la propietat d'**adjacència**.

Codi Binari: *(000, 001, 010, 011, 100, 101, 110, 111)*

Codis de Gray *(000, 001, 011, 010, 110, 111, 101, 100).*