Searching via sorting

Consider the phone book for Addis Ababa. Suppose that it has 1 million names in it. But still, we can find a number easily because it is **alphabetically sorted**.

What would happen if the names were listed in the phone book in random order?

This is true in general - we can find items much faster in arrays that are **sorted**:

(In Python an array and a list are basically the same thing, in other programming languages they can be different.)

```
In [2]: def search(L,item):
            """Search in an unorted array"""
            for i in range(len(L)):
                sys.stdout.write('*')
                if L[i]==item:
                    return i
            return -1
```

```
In [3]: L = range(200)
```

```
In [4]: search(L,100)

        *****************************************************************************
        ************************

Out[4]: 100
```

Can we do it faster using the fact that `L` is **sorted**?

Turns out the answer is **yes**

Binary Search

Input: Sorted array L of length n, item $item$

Output: Index i such that $L[i] == item$ or -1 if no such i exists.

Operation: Check if $L[n/2] > item$.

If YES, then check if $L[n/4] > item$, if NO then check if $L[3n/4] > item$.

If first check was YES and second YES, check if $L[n/8] > item$.

If first check was YES and second NO, check if $L[3n/8] > item$.

If first check was NO and second NO, check if $L[7n/8] > item$.

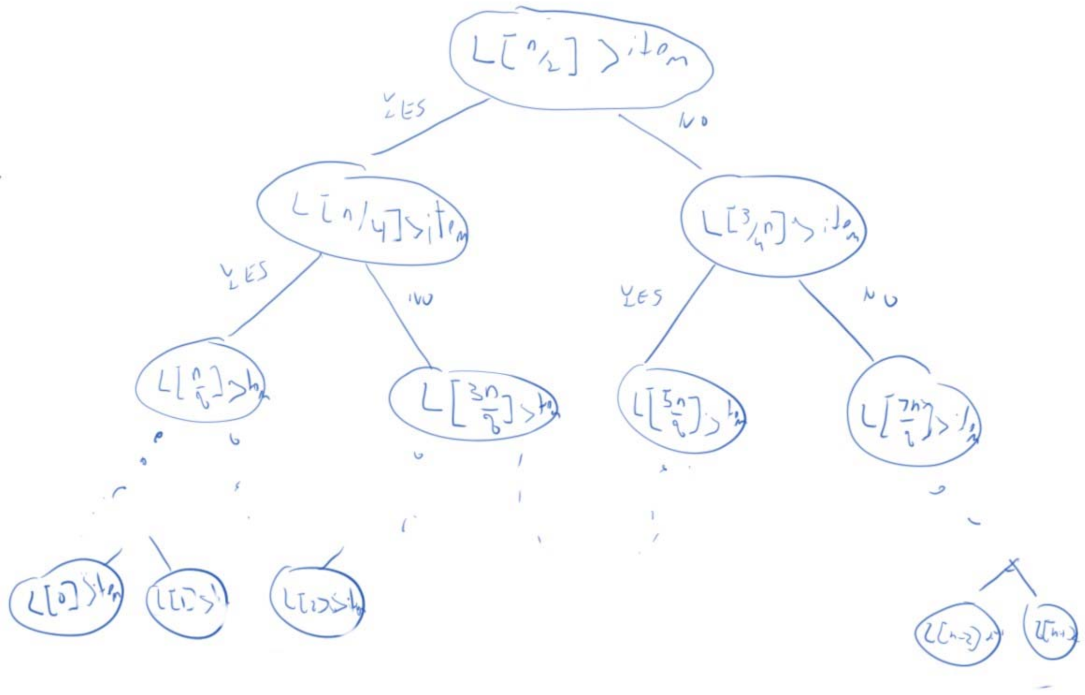
If first check was NO and second YES, check if $L[5n/8] > item$.

....

continue in this way

```
In [5]: #overview of binary search
```

Out[5]:



Binary Search

(a bit more formal operation)

Input: Sorted array L of length n, item $item$

Output: Index i such that $L[i] == item$ or -1 if no such i exists.

Operation:

Check if $L[n/2]

- if YES, continue search in $L[0 : n/2]$
- if NO. continue search in $L[n/2 : n]$

In [6]:
```
def bin_search(L,item):
    sys.stdout.write('*')
    n = len(L)
    if not n:
        return -1
    m = int(n/2)
    if L[m]==item:
        return m
    if L[m]>item:
        return bin_search(L[:m],item)
    res = bin_search(L[m:n],item)
    if res==-1:
        return -1
    return m+res
```

In [7]:
```
search(L,100)
```
```
*****************************************************************************
************************
```
Out[7]:
```
100
```

In [8]:
```
bin_search(L,100)
```
```
********
```
Out[8]:
```
100
```

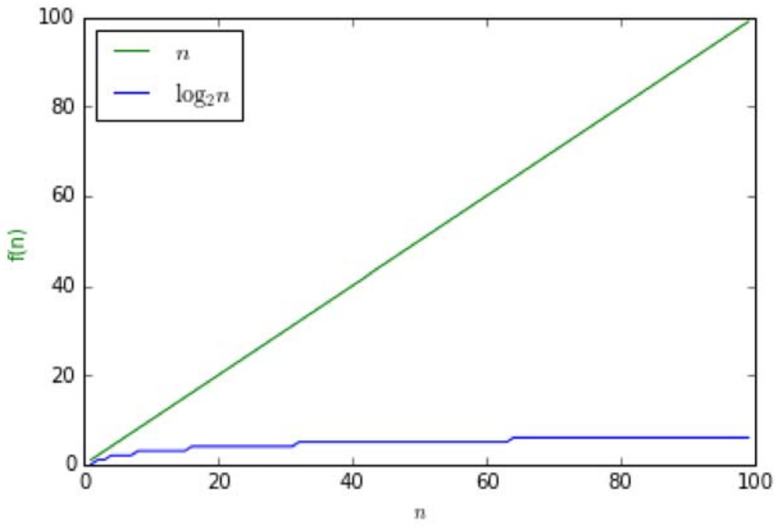
If you run a binary search on a string of length n, then in one step we reduce the problem to a string of length $n/2$, in another step to a string of length $n/4$, and so on.

So the number of steps is the number of items in the sequence $n, n/2, n/4, n/8, \ldots, 1$

In other words, the number of steps binary search takes is the number t such that $n/2^t \leq 1$, which means $t = \rceil log_2 n\rceil \leq \log_2 n + 1$

$\log_2 n$ is much much smaller than n.

In [26]:
```
# compare n with log_2 n
```



In particular, if Facebook wants to search for a user in the data base of 10^9 users, then if they keep the list sorted, they can do it in 30 steps instead of $1,000,000,000$.

For example, Facebook can have a list of all the emails of their users, sorted by their name. Now, given any string `name`, in 30 steps they can find the email corresponding to this user.

Let's be more specific and, since we don't have the list of Facebook users, consider the list of students in this course.

Suppose we have this list of teams in this course:

```
In [16]: groups = ['1:asmare habtemu, biniam kidane, lewi mekonnen', '2:gatlyak chnol-p
at, ochan odoll-oqalla, samuel testage', '3:dawit fikru, kenasa desta, medin s
eid', '4:amina bedri, eman hassen, lidya jegnaw', '5:abreham tuna, haile g/sel
assie, mikiyas legesse', '6:mintesinot tefera, surafel lewtu, tariku erena',
'7:abdurezak temam, adonay geremew, henok mersha', '8:hana alselam-haji, hoodo
 muktar, nasra dayib', '9:hassen ali, kiflom leul, yosef enawgaw', '10:eskedar
 tewabe, tsegereda sebhat, zakira tebarek', '11:helina tewodros, nardos gesses
e', '12:mebrhit girmay, mekides muluneh, nejat beshir', '13:elsaye loha, h/mar
iam shimeles, natinael adelew', '14:betelehem eshetu, bethelhem dessalegn', '1
5:hasane abdi, ibrahim ahmed, nour ahmed', '16:shambel abate, tigabu g/cherko
s, yonatan wesenyelah', '17:haymanot gidena, kumneger worke, tsega hailu', '1
8:mebrahtu lwelo', '19:misgina gebretsadik', '20:hibist wondmeggn, kalkidan mu
luneh, meskerem birhanu', '21:banchiayehu asrat, elsabet bizuneh, yetnayet bir
hanu', '22:aman musa-umare, mohammed nuru, tibebu solomon, zelalem addisu', '2
3:kewser nassir, radiya behredin, urji hussen', '24:adem mohammed, amanuel asf
aw, natol gizaw', '25:mesfin tamiru, shambel a, yospeh tadewos', '26:betelhem
 walelign, eden ketema, hana tariku', '27:ayantu alene, eden teklu, simret tek
align', '28:abinet mulugeta, bezawit h/mariam, yordano jemberu', '29:daniel ha
gos, zelalem amare', '30:wondimu yohannes, yospeh tadewos']
```

We can make it into a list of pairs of the form *(team number, student)*

```
In [17]: pairs = []
         for s in groups:
             i = s.index(":")
             team = int(s[0:i])
             for name in s[i+1:].split(', '):
                 pairs.append([team,name])
         print pairs
```

```
[[1, 'asmare habtemu'], [1, 'biniam kidane'], [1, 'lewi mekonnen'], [2, 'gatl
yak chnol-pat'], [2, 'ochan odoll-oqalla'], [2, 'samuel testage'], [3, 'dawit
 fikru'], [3, 'kenasa desta'], [3, 'medin seid'], [4, 'amina bedri'], [4, 'em
an hassen'], [4, 'lidya jegnaw'], [5, 'abreham tuna'], [5, 'haile g/selassi
e'], [5, 'mikiyas legesse'], [6, 'mintesinot tefera'], [6, 'surafel lewtu'],
 [6, 'tariku erena'], [7, 'abdurezak temam'], [7, 'adonay geremew'], [7, 'hen
ok mersha'], [8, 'hana alselam-haji'], [8, 'hoodo muktar'], [8, 'nasra dayi
b'], [9, 'hassen ali'], [9, 'kiflom leul'], [9, 'yosef enawgaw'], [10, 'esked
ar tewabe'], [10, 'tsegereda sebhat'], [10, 'zakira tebarek'], [11, 'helina t
ewodros'], [11, 'nardos gessese'], [12, 'mebrhit girmay'], [12, 'mekides mulu
neh'], [12, 'nejat beshir'], [13, 'elsaye loha'], [13, 'h/mariam shimeles'],
 [13, 'natinael adelew'], [14, 'betelehem eshetu'], [14, 'bethelhem dessaleg
n'], [15, 'hasane abdi'], [15, 'ibrahim ahmed'], [15, 'nour ahmed'], [16, 'sh
ambel abate'], [16, 'tigabu g/cherkos'], [16, 'yonatan wesenyelah'], [17, 'ha
ymanot gidena'], [17, 'kumneger worke'], [17, 'tsega hailu'], [18, 'mebrahtu
 lwelo'], [19, 'misgina gebretsadik'], [20, 'hibist wondmeggn'], [20, 'kalkid
an muluneh'], [20, 'meskerem birhanu'], [21, 'banchiayehu asrat'], [21, 'elsa
bet bizuneh'], [21, 'yetnayet birhanu'], [22, 'aman musa-umare'], [22, 'moham
med nuru'], [22, 'tibebu solomon'], [22, 'zelalem addisu'], [23, 'kewser nass
ir'], [23, 'radiya behredin'], [23, 'urji hussen'], [24, 'adem mohammed'], [2
4, 'amanuel asfaw'], [24, 'natol gizaw'], [25, 'mesfin tamiru'], [25, 'shambe
l a'], [25, 'yospeh tadewos'], [26, 'betelhem walelign'], [26, 'eden ketem
a'], [26, 'hana tariku'], [27, 'ayantu alene'], [27, 'eden teklu'], [27, 'sim
ret tekalign'], [28, 'abinet mulugeta'], [28, 'bezawit h/mariam'], [28, 'yord
ano jemberu'], [29, 'daniel hagos'], [29, 'zelalem amare'], [30, 'wondimu yoh
annes'], [30, 'yospeh tadewos']]
```

Now we can sort these pairs based on student name:

In [18]:
```
def name(pair):
    return pair[1]

pairs = sorted(pairs,key=name)
print pairs
```

```
[[7, 'abdurezak temam'], [28, 'abinet mulugeta'], [5, 'abreham tuna'], [24,
 'adem mohammed'], [7, 'adonay geremew'], [22, 'aman musa-umare'], [24, 'aman
uel asfaw'], [4, 'amina bedri'], [1, 'asmare habtemu'], [27, 'ayantu alene'],
 [21, 'banchiayehu asrat'], [14, 'betelehem eshetu'], [26, 'betelhem walelig
n'], [14, 'bethelhem dessalegn'], [28, 'bezawit h/mariam'], [1, 'biniam kidan
e'], [29, 'daniel hagos'], [3, 'dawit fikru'], [26, 'eden ketema'], [27, 'ede
n teklu'], [21, 'elsabet bizuneh'], [13, 'elsaye loha'], [4, 'eman hassen'],
 [10, 'eskedar tewabe'], [2, 'gatlyak chnol-pat'], [13, 'h/mariam shimeles'],
 [5, 'haile g/selassie'], [8, 'hana alselam-haji'], [26, 'hana tariku'], [15,
 'hasane abdi'], [9, 'hassen ali'], [17, 'haymanot gidena'], [11, 'helina tew
odros'], [7, 'henok mersha'], [20, 'hibist wondmeggn'], [8, 'hoodo muktar'],
 [15, 'ibrahim ahmed'], [20, 'kalkidan muluneh'], [3, 'kenasa desta'], [23,
 'kewser nassir'], [9, 'kiflom leul'], [17, 'kumneger worke'], [1, 'lewi meko
nnen'], [4, 'lidya jegnaw'], [18, 'mebrahtu lwelo'], [12, 'mebrhit girmay'],
 [3, 'medin seid'], [12, 'mekides muluneh'], [25, 'mesfin tamiru'], [20, 'mes
kerem birhanu'], [5, 'mikiyas legesse'], [6, 'mintesinot tefera'], [19, 'misg
ina gebretsadik'], [22, 'mohammed nuru'], [11, 'nardos gessese'], [8, 'nasra
 dayib'], [13, 'natinael adelew'], [24, 'natol gizaw'], [12, 'nejat beshir'],
 [15, 'nour ahmed'], [2, 'ochan odoll-oqalla'], [23, 'radiya behredin'], [2,
 'samuel testage'], [25, 'shambel a'], [16, 'shambel abate'], [27, 'simret te
kalign'], [6, 'surafel lewtu'], [6, 'tariku erena'], [22, 'tibebu solomon'],
 [16, 'tigabu g/cherkos'], [17, 'tsega hailu'], [10, 'tsegereda sebhat'], [2
3, 'urji hussen'], [30, 'wondimu yohannes'], [21, 'yetnayet birhanu'], [16,
 'yonatan wesenyelah'], [28, 'yordano jemberu'], [9, 'yosef enawgaw'], [25,
 'yospeh tadewos'], [30, 'yospeh tadewos'], [10, 'zakira tebarek'], [22, 'zel
alem addisu'], [29, 'zelalem amare']]
```

In [19]:
```
teams, names = zip(*pairs)
print names
```

```
('abdurezak temam', 'abinet mulugeta', 'abreham tuna', 'adem mohammed', 'adon
ay geremew', 'aman musa-umare', 'amanuel asfaw', 'amina bedri', 'asmare habte
mu', 'ayantu alene', 'banchiayehu asrat', 'betelehem eshetu', 'betelhem walel
ign', 'bethelhem dessalegn', 'bezawit h/mariam', 'biniam kidane', 'daniel hag
os', 'dawit fikru', 'eden ketema', 'eden teklu', 'elsabet bizuneh', 'elsaye l
oha', 'eman hassen', 'eskedar tewabe', 'gatlyak chnol-pat', 'h/mariam shimele
s', 'haile g/selassie', 'hana alselam-haji', 'hana tariku', 'hasane abdi', 'h
assen ali', 'haymanot gidena', 'helina tewodros', 'henok mersha', 'hibist won
dmeggn', 'hoodo muktar', 'ibrahim ahmed', 'kalkidan muluneh', 'kenasa desta',
 'kewser nassir', 'kiflom leul', 'kumneger worke', 'lewi mekonnen', 'lidya je
gnaw', 'mebrahtu lwelo', 'mebrhit girmay', 'medin seid', 'mekides muluneh',
 'mesfin tamiru', 'meskerem birhanu', 'mikiyas legesse', 'mintesinot tefera',
 'misgina gebretsadik', 'mohammed nuru', 'nardos gessese', 'nasra dayib', 'na
tinael adelew', 'natol gizaw', 'nejat beshir', 'nour ahmed', 'ochan odoll-oqa
lla', 'radiya behredin', 'samuel testage', 'shambel a', 'shambel abate', 'sim
ret tekalign', 'surafel lewtu', 'tariku erena', 'tibebu solomon', 'tigabu g/c
herkos', 'tsega hailu', 'tsegereda sebhat', 'urji hussen', 'wondimu yohanne
s', 'yetnayet birhanu', 'yonatan wesenyelah', 'yordano jemberu', 'yosef enawg
aw', 'yospeh tadewos', 'yospeh tadewos', 'zakira tebarek', 'zelalem addisu',
 'zelalem amare')
```

And define binary search to use this key too:

```
In [20]: def bin_search_name(L,s_name):
             n = len(L)
             if not n:
                 return -1
             m = int(n/2)
             if name(L[m])==s_name:
                 return m
             if name(L[m])>s_name:
                 return bin_search_name(L[:m],s_name)
             res = bin_search_name(L[m+1:n],s_name)
             if res==-1:
                 return -1
             return m+res+1
```

```
In [21]: idx = bin_search_name(pairs,'Abinet Mulugeta')
         print "Group number", pairs[idx][0]

         Group number 29
```

```
In [22]: idx = bin_search_name(pairs,'Yonatan Wesenyelah')
         print "Group number", pairs[idx][0]

         Group number 29
```

Dictionaries

This kind of tasks - storing information that you want to access using some *key*, is so common that python has a special data structure for it called a **dictionary**

```
In [132]: groups_dict = { 'adem mohammed': 24, 'samuel testage': 2, 'Kalkidan Muluneh':
          20 }
```

```
In [1]: groups_dict['samuel testage']

        2
```

We can add to `groups_dict` *all* the pairs as follows:

```
In [156]: for pair in pairs:
              groups_dict[pair[1]] = int(pair[0])
          print groups_dict
```

```
{'betelhem walelign': 26, 'Eman Hassen': 4, 'Aman Musa-Umare': 22, 'Haymanot
 Gidena': 17, 'Hassen Ali': 9, 'wondimu yohannes': 30, 'Helina Tewodros': 11,
 'Kewser Nassir': 23, 'zelalem amare': 29, 'elsaye loha': 13, 'mohammed nur
u': 22, 'tariku erena': 6, 'Ibrahim Ahmed': 15, 'Abdurezak Temam': 7, 'mekide
s muluneh': 12, 'simret tekalign': 27, 'surafel lewtu': 6, 'natol gizaw': 24,
 'Adonay Geremew': 7, 'tigabu g/cherkos': 16, 'kewser nassir': 23, 'lidya jeg
naw': 4, 'yonatan wesenyelah': 16, 'haile g/selassie': 5, 'Bezawit H/Mariam':
 28, 'daniel hagos': 29, 'Mesfin Tamiru': 25, 'Yetnayet Birhanu': 21, 'eden k
etema': 26, 'adonay geremew': 7, 'Lewi Mekonnen': 1, 'kumneger worke': 17, 'B
anchiayehu Asrat': 21, 'Natol Gizaw': 24, 'mesfin tamiru': 25, 'Amina Bedri':
 4, 'Hana Alselam-Haji': 8, 'Samuel Testage': 2, 'Wondimu Yohannes': 30, 'med
in seid': 3, 'Yosef Enawgaw': 9, 'Ochan Odoll-Oqalla': 2, 'hoodo muktar': 8,
 'mikiyas legesse': 5, 'asmare habtemu': 1, 'adem mohammed': 24, 'samuel test
age': 2, 'Kalkidan Muluneh': 20, 'biniam kidane': 1, 'h/mariam shimeles': 13,
 'radiya behredin': 23, 'yospeh tadewos': 30, 'Daniel Hagos': 29, 'haymanot g
idena': 17, 'henok mersha': 7, 'hassen ali': 9, 'Eden Teklu': 27, 'eden tekl
u': 27, 'Nour Ahmed': 15, 'hana alselam-haji': 8, 'nejat beshir': 12, 'Mikiya
s Legesse': 5, 'Hasane Abdi': 15, 'Nardos Gessese': 11, 'aman musa-umare': 2
2, 'natinael adelew': 13, 'Hana Tariku': 26, 'Mekides Muluneh': 12, 'Meskerem
 Birhanu': 20, 'Yospeh Tadewos': 30, 'Kumneger Worke': 17, 'Lidya Jegnaw': 4,
 'shambel abate': 16, 'amina bedri': 4, 'Betelhem Walelign': 26, 'Gatlyak Chn
ol-Pat': 2, 'urji hussen': 23, 'dawit fikru': 3, 'Mintesinot Tefera': 6, 'hib
ist wondmeggn': 20, 'nour ahmed': 15, 'Medin Seid': 3, 'helina tewodros': 11,
 'Zelalem Addisu': 22, 'betelehem eshetu': 14, 'abreham tuna': 5, 'Tariku Ere
na': 6, 'hasane abdi': 15, 'nardos gessese': 11, 'tsega hailu': 17, 'bezawit
 h/mariam': 28, 'Adem Mohammed': 24, 'yordano jemberu': 28, 'H/Mariam Shimele
s': 13, 'Abinet Mulugeta': 28, 'tsegereda sebhat': 10, 'Nasra Dayib': 8, 'Urj
i Hussen': 23, 'Yordano Jemberu': 28, 'kalkidan muluneh': 20, 'amanuel asfa
w': 24, 'Eden Ketema': 26, 'bethelhem dessalegn': 14, 'Mebrhit Girmay': 12,
 'Abreham Tuna': 5, 'kiflom leul': 9, 'abdurezak temam': 7, 'Hibist Wondmegg
n': 20, 'Simret Tekalign': 27, 'lewi mekonnen': 1, 'Shambel A': 25, 'meskerem
 birhanu': 20, 'mebrahtu lwelo': 18, 'Hoodo Muktar': 8, 'misgina gebretsadi
k': 19, 'ochan odoll-oqalla': 2, 'ayantu alene': 27, 'eman hassen': 4, 'Zelal
em Amare': 29, 'Dawit Fikru': 3, 'Biniam Kidane': 1, 'Nejat Beshir': 12, 'Zak
ira Tebarek': 10, 'Asmare Habtemu': 1, 'tibebu solomon': 22, 'Bethelhem Dessa
legn': 14, 'Amanuel Asfaw': 24, 'Shambel Abate': 16, 'Kiflom Leul': 9, 'Ayant
u Alene': 27, 'Haile G/Selassie': 5, 'kenasa desta': 3, 'Surafel Lewtu': 6,
 'zakira tebarek': 10, 'Elsabet Bizuneh': 21, 'Henok Mersha': 7, 'Tsega Hail
u': 17, 'Tsegereda Sebhat': 10, 'nasra dayib': 8, 'Betelehem Eshetu': 14, 'mi
ntesinot tefera': 6, 'Yonatan Wesenyelah': 16, 'Eskedar Tewabe': 10, 'Misgina
 Gebretsadik': 19, 'abinet mulugeta': 28, 'yetnayet birhanu': 21, 'Elsaye Loh
a': 13, 'yosef enawgaw': 9, 'mebrhit girmay': 12, 'Mohammed Nuru': 22, 'Tibeb
u Solomon': 22, 'eskedar tewabe': 10, 'gatlyak chnol-pat': 2, 'ibrahim ahme
d': 15, 'zelalem addisu': 22, 'Natinael Adelew': 13, 'hana tariku': 26, 'sham
bel a': 25, 'Mebrahtu Lwelo': 18, 'Kenasa Desta': 3, 'Radiya Behredin': 23,
 'elsabet bizuneh': 21, 'banchiayehu asrat': 21, 'Tigabu G/Cherkos': 16}
```

Are dictionaries implemented using sorted arrays?

Problem: What if you add a new element - do you need to re-sort the whole array?

Two main solutions: *Binary search trees* and *hash tables*.

Python uses hash tables and this is what we explain next.

Hash tables*

Idea: Design a function `h(s)` such that for every string `s` , `h(s)` is a number between 0 and n. Then, we can have a list `L` of length n such that `L[h(s)]` will be the group of student `s`

So, to get the group of the student with name `name`, all we need to do is to compute `h(name)` and then we can get this student in only one step.

Problems:

- How do you find such a function?
- What do you do if you have two different students of with names `name1` and `name2` such that `h(name1)==h(name2)`?
- How small can we make n? Note that it costs us in computer *memory* to make n too big.

Let's start with the first problem: we want to find a function h that takes every string s to a somewhat "random number" between 1 and n. (in our case $n = 81$)

One simple function is the following: treat each letter as a number from 1 to 26 and add all the letters in the name modulo n .

(note that this is a simple function that works well sometimes but not always, and in particular will always map "boaz" and "azbo" to the same number; there are better "hash functions" that are used in Python and other systems.)

```
In [9]:  def letter_to_number(c):
              if c==' ':
                  return 27
              if c=='-':
                  return 28
              return 1+ord(c)-ord('a')

          letter_to_number('c')

Out[9]:   3
```

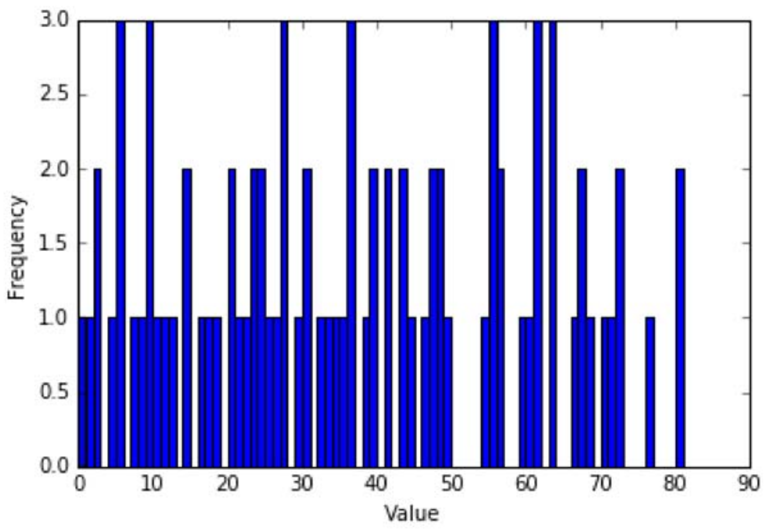
```
In [10]: def h(s,n):
             res = 0
             for c in s.lower():
                 res += letter_to_number(c)
             return res % n
```

```
In [14]: h("boaz barak",83)

Out[14]: 21
```

We don't have the list of all Facebook users, so let's test how well it works for the students in this class:

```
In [23]: len(pairs)

Out[23]: 83
```

```
In [24]: integer_hist([h(pair[1],83) for pair in pairs])
```



We see that the function is "almost" good, in that most places only have one student matched to it, but several places have two students and a few have three students.

So now we can have a list `groups_list` of length 83, where for every `i`, `groups_list[i]` will contains the list of all pairs corresponding to the students with name `s` such that `h(s)==i`.

For every `i`, the list `groups_list[i]` will have at most three pairs.

So, if we want to get the group of a student with name `s` we need to do it in at most four steps:

- We let `L=groups_list[h(s)]`
- Then we scan this short list `L` to find the pair of the form `[ t , s]`

Summary of data structures

Often the right data structure can make all the difference:

Data structure	Get(key).....	Insert(key)..	Other properties
Unsorted list	n	< 10 (*)	Supports any objects
Sorted array	$\log n$	n	Supports range queries
Search trees	$\log n$	$\log n$	Support range queries
Hash table	< 10	< 10	Supports non-comparable keys

Note: Data structures is a *huge* topic and if you study more computer science you will hear about more concepts such as stacks, queues, heaps, and many more.

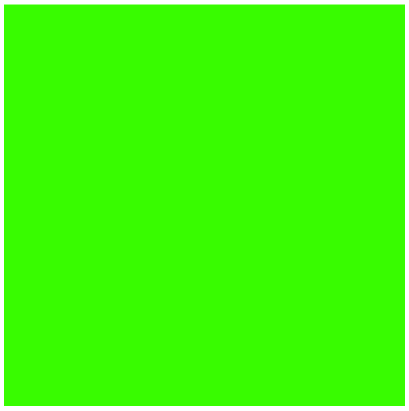
Graphics on a computer

The image you see is composed of about 1,000,000 little dots known as *pixels* (1024×768).

Each pixel can be set to a different color, and that produces the image.

All colors are obtained by mixing red, green and blue

```
In [30]: demo_RGB()
```



Python allows us to take an array of color values for the pixels and plot it on the screen. To make things easier, we implemented some helper functions to do it:

- `color(red,green,blue)`: takes three numbers and simply returns a list of these three numbers, but it can also be called with named parameters and has defeault values.
- `empty_screen(height,width)`: returns an `width` $\times$ `height` array `s` (namely a list of `width` lists, each of them is of length `height`). For every x between 0 and `width` and y between 0 and `height`, `s[x][y] = color(255,255,255)`.
- `plot_array(s)`: plots the array `s` on the screen where `s[0][0]` corresponds to the bottom left corner and `s[width][length]` corresponds to the top right corner.

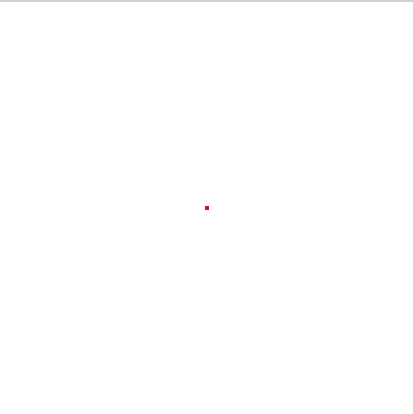
We will now demonstrate how to use these functions:

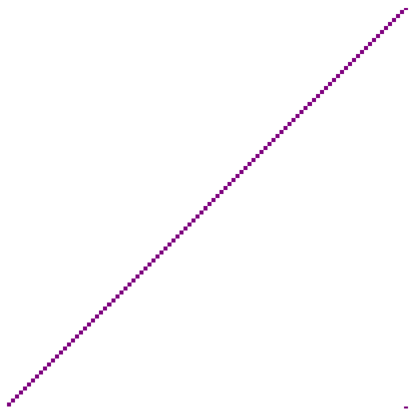
```
In [82]: color(20,30,40)

Out[82]: (20, 30, 40)
```

```
In [83]: color(blue=255)

Out[83]: (0, 0, 255)
```

```
In [84]: s = empty_screen(100,100)
```

```
In [85]: plot_array(s)
```

In [86]:
```
s[50][50] = color(red=255,blue=0,green=0)
plot_array(s)
```


In [87]:
```
import time
for i in range(100):
    s[i][i]=color(red=128,blue=128)
    plot_array(s)
    # time.sleep(0.01)
```


In [88]:
```
import math
def sine(angle):
    return math.sin((angle/360.0)*2*math.pi)
def cosine(angle):
    return math.cos((angle/360.0)*2*math.pi)
```

In [125]:
```
def cannon(angle,speed,time, gravity=9.8):
    x = speed*time*cosine(angle)
    y = speed*time*sine(angle) -(gravity/2.0)*(time**2)
    return round(x,3),round(y,3)
```

In [110]:
```
cannon(45,10,100,gravity=0)
```
Out[110]:
```
(707.107, 707.107)
```

In [111]:
```
cannon(45,10,200,gravity=0)
```
Out[111]:
```
(1414.214, 1414.214)
```

```
In [112]: cannon(30,10,100,gravity=0)
Out[112]: (866.025, 500.0)
```

```
In [113]: cannon(70,10,100,gravity=0)
Out[113]: (342.02, 939.693)
```

```
In [123]: cannon(45,100,10)
Out[123]: (707.107, 217.107)
```

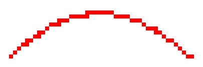
```
In [126]: cannon(45,100,20)
Out[126]: (1414.214, -545.786)
```

```
In [120]: cannon(30,100,10)
Out[120]: (866.025, 10.0)
```

```
In [121]: cannon(60,100,10)
Out[121]: (500.0, 376.025)
```

```
In [90]: def draw_cannon(angle,speed):
             s = empty_screen(100,100)
             x =0
             y =0
             t=0.0
             while x<100 and y>=0 and y<100:
                 s[x][y] = color(red=255)
                 (x,y) = cannon(angle,speed,t)
                 x= int(x)
                 y = int(y)
                 t += 1.0/speed
                 plot_array(s)
```

```
In [92]: draw_cannon(45,30)
```



Quiz tomorrow

- One hour on the computer.
- Be here before **9:15am**: half the people will take the quiz, half will do lab work, and then switch.
- Some questions would be easier and some harder - *don't feel bad if you can't solve them all!* (or even most)
- quiz is mostly for us, so we know what concepts you know and what concepts you don't.
- some questions would be very similar to your labworks.
- if you didn't complete the labworks: talk to your friends that did complete it, and make sure you understand the solutions.

Labwork

Choose one of two projects to complete: (if you finish one, you can also try the other)

In these projects you can use all the functions that you have built in previous lab works.

Project 1: Linear equations solver

Write a function `equation_solver` that asks a user for the number n of equations and variables, and then for n equations of the form `10y-0.5x+25z + 50 = 0` and prints a solution of the form `x=10.0, y=0.5, z=2.0`

The function can use the function `solve` we've seen before, any helper functions you already made or new ones. In particular, write the helper functions

- `parse_equation(s)` that takes a string representing an equation in n variables and returns a list of $n+1$ numbers that represents the coefficients for all variables (in alphabetical order) and the constant coefficient.
- `get_variables(user_equations)` that takes a list of strings corresponding to equations and returns a sorted list of all variables appearing in those equations. Each variable should only appear once so if the equations are in n variables then the length of the list that is returned should be n.

The function `equation_solver` itself needs to be of the form below.

bonus: handle equations with missing variables, equations where the right hand side is not just `= 0`, equations that have infinitely many or no solutions (for the latter one you might need to modify `solve`)

In [128]:

```
### HELPER FUNCTIONS:

def make_first_coeff_nonzero_general(eqs):
    for i in range(len(eqs)):
        if eqs[i][0]:
            eqs[0], eqs[i] = eqs[i], eqs[0]
            return
    sys.exit("oops:  all first coefficients are zero")
    return

def multiply_equation(eq,num):
    """Multiply all coefficients of equation eq by number num.
       Return result"""
    res = []
    for x in eq:
        res += [x*num]
    return res

def add_equations(eq1,eq2):
    """Add eq1 and eq2. Return result"""
    res = []
    for i in range(len(eq1)):
        res.append(eq1[i]+eq2[i])
    return res

def solve(eqs):
    n = len(eqs)
    make_first_coeff_nonzero_general(eqs)  # make 1st coef of 1st equation non
zero
    eqs[0] = multiply_equation(eqs[0],1/eqs[0][0])
    # make 1st coef of 1st  equation equal 1

    for i in range(1,n-1):
        eqs[i] = add_equations(eqs[i],multiply_equation(eqs[0],-eqs[i][0])) #
 zero out first coefficient in eqs 1,2
    # make 1st coef of 2nd .. n-th equation equal zero

    rest_equations = []
    for i in range(1,n):
        rest_equations.append(eqs[i][1:n+1])

    solutions = solve(rest_equations)
    # solve remainder of  equations for remainder of  variables

    x =  - eqs[0][n]
    for i in range(1,n):
        x -= eqs[0][i]*solutions[i-1]
    # solve 1st variable using solution for 2nd and 3rd variable

    return [x] + solutions
```

In [129]:
```
def get_variables(user_equations):
    """
    Gets list of strings including some equations and returns a sorted list of
 all variables that
    appear in these equations.
    You can assume all variables are a single lower-case letter.
    """

    # YOUR CODE HERE
    # needs to return a list
```

In [ ]:
```
def parse_equation(s,variables):
    """
    Gets string input representing an equation in the variables in the sorted
 list `variables`
    and outputs a list `eq` of length `len(variables)+1` that contains the coe
fficents for
    each variable (in alphabetical order) and the coefficient for the constant
 term.
    You can assume that each variable is a single lower-case letter
    """

    # YOUR CODE HERE
    # needs to return a list of length len(variables)+1
```

In [ ]:
```
def equation_solver():
    n = int(raw_input("Enter the number of equations / variables"))

    user_equations = []
    for i in range(n):
        s = raw_input("Enter equation number "+str(i)+":")
        user_equations.append(s)

    variables = get_variables(user_equations) # you need to write get_variable
s
    eqs  = []
    for s in user_equations:
        eqs.append(parse_equation(s,variables)) # you need to write parse_equa
tions

    solutions = solve(eqs)

    for i in len(solutions):
        print variables[i]+ " = "+str(solutions[i])
```

Project 2: Cannon game

Prepare a game for two players that will work as follows:

Each player chooses a *position*, *angle* and *speed* for their cannon.

Then the two cannons shoot and each player scores if they hit the other player's cannon.

```
In [ ]: def cannon_game(width,height):
            angle1 = int(raw_input("Player 1: enter your angle"))
            speed1 = int(raw_input("Player 1: enter your speed"))
            location1 = int(raw_input("Player 1: enter your location"))

            # TO BE COMPLETED
```