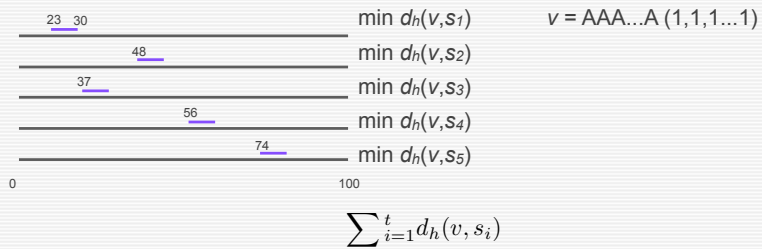


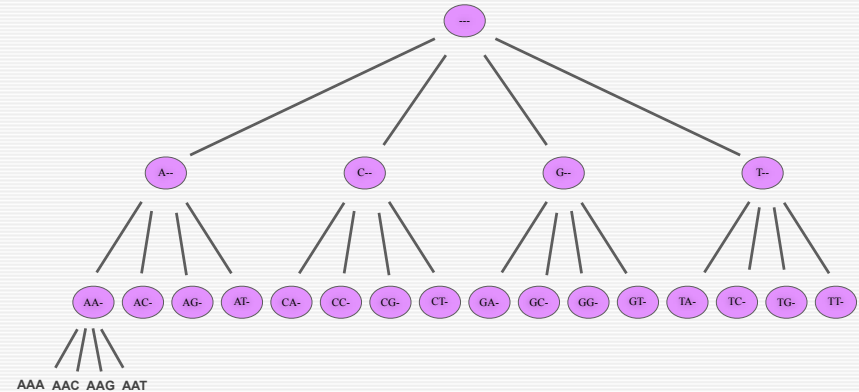
# MEDIAN STRING (TAKE 2)



evaluate: (1,1,1,...1) .... (4,4,4,...4)

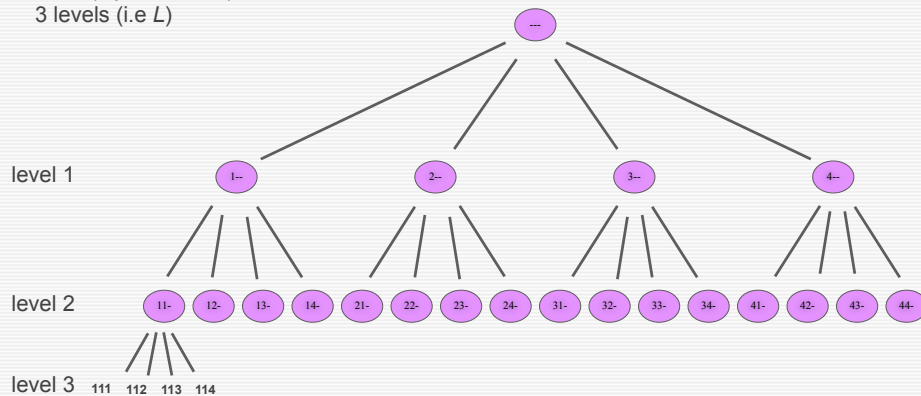
# ORGANIZE THE SEARCH

3-mer  
k = 4 (alphabet size)



# STEP 1: ORGANIZE THE SEARCH

3-mer  
k = 4 (alphabet size)  
3 levels (i.e L)



# NEXTLEAF

```

NEXTLEAF(a,L,k)
  for i = L to 1
    if a_i < k
      a_i = a_i + 1
      return a
    a_i = 1
  return a
    
```

a = array of integers which represent motif  
L = length of motif (L=3)  
k = alphabet size (k=4)

we are at leaf a = (1,1,1)

```

NEXTLEAF(a=(1,1,1),L=3,k=4)
  for i = 3 to 1
    if a_i < 4
      a_i = a_i + 1
      return a
    a_i = 1
  return a
return (1,1,2)
    
```

we are at leaf a = (1,1,4)

```

NEXTLEAF(a=(1,1,4),L=3,k=4)
  for i = 3 to 1
    if a_i < 4
      a_i = a_i + 1
      return a
    a_i = 1
  return a
return (1,1,1)
    
```



# NEXTVERTEX

NEXTVERTEX(a,i,L,k)

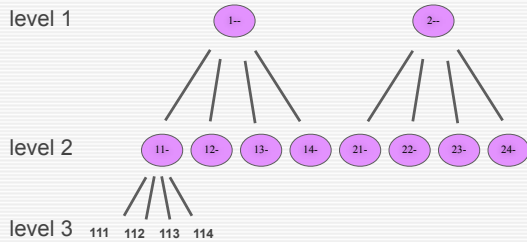
```

if i < L
  ai+1 = 1
  return (a,i+1)
else
  for j = L to 1
    if aj < k
      aj = aj + 1
      return (a,j)
  return (a,0)
  
```

a = array of integers which represent motif  
 i = level of the tree  
 L = length of motif (L=3)  
 k = alphabet size (k=4)

```

if a = (1,-,-)
  NEXTVERTEX(a=(1,-,-),i=1,L=3,k=4)
  if i < 3
    ai+1 = 1
    return (a=(1,1,-),i+1=2)
  else
    for j = L to 1
      if aj < k
        aj = aj + 1
        return (a,j)
    return (a,0)
  
```



# NEXTVERTEX

NEXTVERTEX(a,i,L,k)

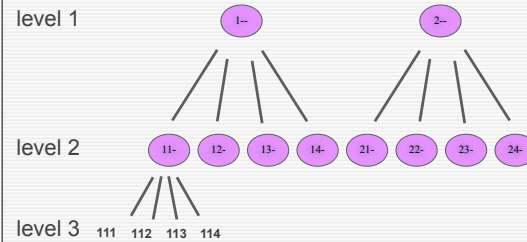
```

if i < L
  ai+1 = 1
  return (a,i+1)
else
  for j = L to 1
    if aj < k
      aj = aj + 1
      return (a,j)
  return (a,0)
  
```

a = array of integers which represent motif  
 i = level of the tree  
 L = length of motif (L=3)  
 k = alphabet size (k=4)

```

if a = (1,1,1)
  NEXTVERTEX(a=(1,1,1),i=3,L=3,k=4)
  if i < 3
    ai+1 = 1
    return (a,i+1)
  else
    for j = 3 to 1
      if aj < k
        aj = aj + 1
        return (a=(1,1,2),j=3)
    return (a,0)
  
```



# NEXTVERTEX

NEXTVERTEX(a,i,L,k)

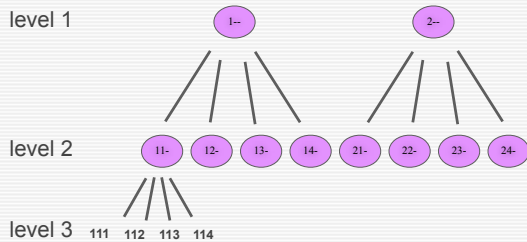
```

if i < L
  ai+1 = 1
  return (a,i+1)
else
  for j = L to 1
    if aj < k
      aj = aj + 1
      return (a,j)
  return (a,0)
  
```

a = array of integers which represent motif  
 i = level of the tree  
 L = length of motif (L=3)  
 k = alphabet size (k=4)

```

if a = (1,1,4)
  NEXTVERTEX(a=(1,1,4),i=3,L=3,k=4)
  if i < 3
    ai+1 = 1
    return (a,i+1)
  else
    for j = 3 to 1
      if aj < k
        aj = aj + 1
        return (a=(1,2,4),j=2)
    return (a,0)
  
```



# NEXTVERTEX

NEXTVERTEX(a,i,L,k)

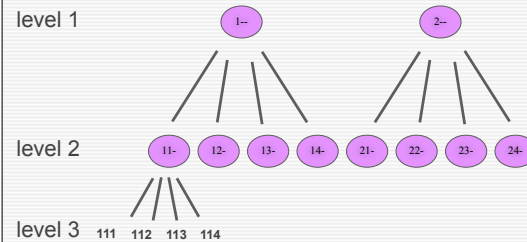
```

if i < L
  ai+1 = 1
  return (a,i+1)
else
  for j = L to 1
    if aj < k
      aj = aj + 1
      return (a,j)
  return (a,0)
  
```

a = array of integers which represent motif  
 i = level of the tree  
 L = length of motif (L=3)  
 k = alphabet size (k=4)

```

if a = (1,2,4) > (1,2,-)
  NEXTVERTEX(a=(1,2,4),i=2,L=3,k=4)
  if i < 3
    ai+1 = 1
    return (a=(1,2,1),3)
  else
    for j = 3 to 1
      if aj < k
        aj = aj + 1
        return (a,j)
    return (a,0)
  
```

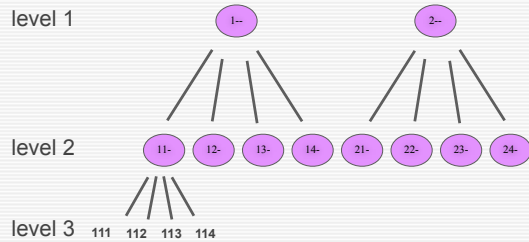


# BYPASS

```

BYPASS(a,i,L,k)
  for j = i to 1
    if aj < k
      aj = aj + 1
      return (a,j)
  return (a,0)
    
```

**a** = array of integers which represent motif  
*i* = level of the tree  
*L* = length of motif (*L*=3)  
*k* = alphabet size (*k*=4)  
 skip the subtree rooted at vertex (a,i)



```

BYPASS(a=(1,-,-),i=1,L=3,k=4)
  for j = i to 1
    if aj < k
      aj = aj + 1
      return (a=(2,-,-),j)
  return (a,0)
    
```

# BRANCH & BOUND MEDIAN SEARCH

BRANCHANDBOUNDMEDIANSEARCH(*DNA,t,n,l*)

**v** = (1,1,...,1) // i.e AA...A

*bestDistance* = 1e+27

*i* = 1

**while** *i* > 0

**if** *i* < *l*

*prefix* = nucleotide string for (*v*<sub>1</sub>,*v*<sub>2</sub>,...*v*<sub>*i*</sub>)

*optimisticDistance* = TOTALDISTANCE(*prefix*,*DNA*)

**if** *optimisticDistance* > *bestDistance*

      (**v**,*i*) = BYPASS (**v**,*i*,*l*,4)

**else**

      (**v**,*i*) = NEXTVERTEX(**v**,*i*,*l*,4)

**else**

*word* = nucleotide string for (*v*<sub>1</sub>,*v*<sub>2</sub>,...*v*<sub>*i*</sub>)

**if** TOTALDISTANCE(*word*,*DNA*) < *bestDistance*

*bestDistance* = TOTALDISTANCE(*word*,*DNA*)

*bestWord* = *word*

      (**v**,*i*) = NEXTVERTEX(**v**,*i*,*l*,4)

**return** *bestWord*

