

* Activity Selection

Problem

$$\begin{bmatrix} n = r - p + 1 \\ n_1 = q - p + 1 \\ n_2 = r - q \end{bmatrix} \quad n_1 + n_2 = n$$

RT analysis: Merge algorithmAlgorithm *Merge*(A, p, q, r)

- 1: Copy $A[p..q]$ into $L[1..q - p + 1]$
- 2: Copy $A[q + 1..r]$ into $R[1..r - q]$
- 3: Set sentinels $L[q - p + 2] = R[r - q + 1] = \infty$
- 4: $i \leftarrow 1; \quad j \leftarrow 1$
- 5: **for** $k \leftarrow p$ **to** r **do**
- 6: **if** $L[i] \leq R[j]$ **then**
- 7: $A[k] \leftarrow L[i]; \quad i \leftarrow i + 1$
- 8: **else**
- 9: $A[k] \leftarrow R[j]; \quad j \leftarrow j + 1$

* counting sort
→ value of k ?

$$4n \left[\begin{array}{c} 1 \\ 3 \left[\begin{array}{c} 1 \\ 2 \\ 1 \\ 2 \end{array} \right] \right]$$

$$n_1 + n_2 + 3 + 4n \\ = 3 + 5n \\ O(n)$$

$$n = r - p + 1$$

RT analysis: Partition algorithm

- 1: $i \leftarrow \text{Random}(p, r)$
- 2: Exchange $A[i]$ and $A[r]$
- 3: $x \leftarrow A[r]$
- 4: $i \leftarrow p - 1$
- 5: **for** $j \leftarrow p$ **to** $r - 1$ **do**
- 6: **if** $A[j] \leq x$ **then**
- 7: $i \leftarrow i + 1$
- 8: Exchange $A[i]$ and $A[j]$
- 9: Exchange $A[i + 1]$ and $A[r]$
- 10: **return** $i + 1$

$$5(n-1) \left[\begin{array}{c} 1 \\ 1 \\ 3 \\ 3 \end{array} \right]$$

$$O(n)?$$

$$n = r - p + 1$$

RT analysis: Binary search

1: $BSearch(A, p, r, x)$ // Search for x in sorted array $A[p:r]$
~~2: while $p \leq r$ do~~
 3: $q \leftarrow (p + r) / 2$
 4: if $x < A[q]$ then
 5: $r \leftarrow q - 1$
 6: else if $x > A[q]$ then
 7: $p \leftarrow q + 1$
 8: else
 9: return q // index of x in A
 10: return -1 // return -1 if x is not in A

halted every time
 $8 \log_2 n$
 $O(\log n)$

$n/2$
 $n/4$
 $n/8$
 $n/16$
 $n/32$
 $n/64$
 $n/128$
 $n/256$
 $n/512$
 $n/1024$
 $n/2048$
 $n/4096$
 $n/8192$
 $n/16384$
 $n/32768$
 $n/65536$
 $n/131072$
 $n/262144$
 $n/524288$
 $n/1048576$
 $n/2097152$
 $n/4194304$
 $n/8388608$
 $n/16777216$
 $n/33554432$
 $n/67108864$
 $n/134217728$
 $n/268435456$
 $n/536870912$
 $n/1073741824$
 $n/2147483648$
 $n/4294967296$
 $n/8589934592$
 $n/17179869184$
 $n/34359738368$
 $n/68719476736$
 $n/137438953472$
 $n/274877906944$
 $n/549755813888$
 $n/1099511627776$
 $n/2199023255552$
 $n/4398046511104$
 $n/8796093022208$
 $n/17592186044416$
 $n/35184372088832$
 $n/70368744177664$
 $n/140737488355328$
 $n/281474976710656$
 $n/562949953421312$
 $n/1125899906842624$
 $n/2251799813685248$
 $n/4503599627370496$
 $n/9007199254740992$
 $n/18014398509481984$
 $n/36028797018963968$
 $n/72057594037927936$
 $n/144115188075855872$
 $n/288230376151711744$
 $n/576460752303423488$
 $n/1152921504606846976$
 $n/2305843009213693952$
 $n/4611686018427387904$
 $n/9223372036854775808$
 $n/18446744073709551616$
 $n/36893488147419103232$
 $n/73786976294838206464$
 $n/147573952589676412928$
 $n/295147905179352825856$
 $n/590295810358705651712$
 $n/1180591620717411303424$
 $n/2361183241434822606848$
 $n/4722366482869645213696$
 $n/9444732965739290427392$
 $n/18889465931478580854784$
 $n/37778931862957161709568$
 $n/75557863725914323419136$
 $n/151115727451828646838272$
 $n/302231454903657293676544$
 $n/604462909807314587353088$
 $n/1208925819614629174706176$
 $n/2417851639229258349412352$
 $n/4835703278458516698824704$
 $n/9671406556917033397649408$
 $n/19342813113834066795298816$
 $n/38685626227668133590597632$
 $n/77371252455336267181195264$
 $n/154742504910672534362390528$
 $n/309485009821345068724781056$
 $n/618970019642690137449562112$
 $n/1237940039285380274899124224$
 $n/2475880078570760549798248448$
 $n/4951760157141521099596496896$
 $n/9903520314283042199192993792$
 $n/19807040628566084398385987584$
 $n/39614081257132168796771975168$
 $n/79228162514264337593543950336$
 $n/158456325028528675187087900672$
 $n/316912650057057350374175801344$
 $n/633825300114114700748351602688$
 $n/1267650600228229401496703205376$
 $n/2535301200456458802993406410752$
 $n/5070602400912917605986812821504$
 $n/10141204801825835211973625643008$
 $n/20282409603651670423947251286016$
 $n/40564819207303340847894502572032$
 $n/81129638414606681695789005144064$
 $n/162259276829213363391578010288128$
 $n/324518553658426726783156020576256$
 $n/649037107316853453566312041152512$
 $n/1298074214633706907132624082305024$
 $n/2596148429267413814265248164610048$
 $n/5192296858534827628530496329220096$
 $n/10384593717069655257060992658440192$
 $n/20769187434139310514121985316880384$
 $n/41538374868278621028243970633760768$
 $n/83076749736557242056487941267521536$
 $n/166153499473114484112975882535043072$
 $n/332306998946228968225951765070086144$
 $n/664613997892457936451903530140172288$
 $n/1329227995784915872903807060280344576$
 $n/2658455991569831745807614120560689152$
 $n/5316911983139663491615228241121378304$
 $n/10633823966279326983230456482242756608$
 $n/21267647932558653966460912964485513216$
 $n/42535295865117307932921825928971026432$
 $n/85070591730234615865843651857942052864$
 $n/170141183460469231731687303715884105728$
 $n/340282366920938463463374607431768211456$
 $n/680564733841876926926749214863536422912$
 $n/1361129467683753853853498429727072845824$
 $n/2722258935367507707706996859454145691648$
 $n/5444517870735015415413993718908291383296$
 $n/10889035741470030830827987437816582766592$
 $n/21778071482940061661655974875633165533184$
 $n/43556142965880123323311949751266331066368$
 $n/87112285931760246646623899502532662132736$
 $n/174224571863520493293247799005065324265472$
 $n/348449143727040986586495598010130648530944$
 $n/696898287454081973172991196020261297061888$
 $n/1393796574908163946345982392040522594123776$
 $n/2787593149816327892691964784081045188247552$
 $n/5575186299632655785383929568162090376495104$
 $n/11150372599265311570767859136324180752990208$
 $n/22300745198530623141535718272648361505980416$
 $n/44601490397061246283071436545296723011960832$
 $n/89202980794122492566142873090593446023921664$
 $n/178405961588244985132285746181186892047843328$
 $n/356811923176489970264571492362373784095686656$
 $n/713623846352979940529142984724747568191373312$
 $n/1427247692705959881058285969449495136382746624$
 $n/2854495385411919762116571938898990272765493248$
 $n/5708990770823839524233143877797980545530986496$
 $n/11417981541647679048466287755595961091061972992$
 $n/22835963083295358096932575511191922182123945984$
 $n/45671926166590716193865151022383844364247891968$
 $n/91343852333181432387730302044767688728495783936$
 $n/182687704666362864775460604089535377456991567872$
 $n/365375409332725729550921208179070754913983135744$
 $n/730750818665451459101842416358141509827966271488$
 $n/1461501637330902918203684832716283019655932542976$
 $n/2923003274661805836407369665432566039311865085952$
 $n/5846006549323611672814739330865132078623730171904$
 $n/11692013098647223345629478661730264157247460343808$
 $n/23384026197294446691258957323460528314494920687616$
 $n/46768052394588893382517914646921056628989841375232$
 $n/93536104789177786765035829293842113257979682750464$
 $n/187072209578355573530071658587684226515959365500928$
 $n/374144419156711147060143317175368453031918731001856$
 $n/748288838313422294120286634350736906063837462003712$
 $n/1496577676626844588240573268701473812127674924007424$
 $n/2993155353253689176481146537402947624255349848014848$
 $n/5986310706507378352962293074805895248510699696029696$
 $n/11972621413014756705924586149611790497021399392059392$
 $n/23945242826029513411849172299223580994042798784118784$
 $n/47890485652059026823698344598447161988085597568237568$
 $n/95780971304118053647396689196894323976171195136475136$
 $n/191561942608236107294793378393788647952342390272950272$
 $n/383123885216472214589586756787577295904684780545900544$
 $n/766247770432944429179173513575154591809369561091801088$
 $n/1532495540865888858358347027150309183618739122183602176$
 $n/3064991081731777716716694054300618367237478244367204352$
 $n/6129982163463555433433388108601236734474956488734408704$
 $n/12259964326927110866866776217202473468949912977468817408$
 $n/24519928653854221733733552434404946937899825954937634816$
 $n/49039857307708443467467104868809893875799651909875269632$
 $n/98079714615416886934934209737619787751599303819750539264$
 $n/196159429230833773869868419475239575503198607639501078528$
 $n/392318858461667547739736838950479151006397215279002157056$
 $n/784637716923335095479473677900958302012794430558004314112$
 $n/1569275433846670190958947355801916604025588861116008628224$
 $n/3138550867693340381917894711603833208051177722232017256448$
 $n/6277101735386680763835789423207666416102355444464034512896$
 $n/12554203470773361527671578846415332832204710888928069025792$
 $n/25108406941546723055343157692830665664409421777856138051584$
 $n/50216813883093446110686315385661331328818843555712276103168$
 $n/100433627766186892221372630771322662657637687111424552206336$
 $n/200867255532373784442745261542645325315275374222849104412672$
 $n/401734511064747568885490523085290650630550748445698208825344$
 $n/803469022129495137770981046170581301261101496891396417650688$
 $n/1606938044258990275541962092341162602522202993782792835301376$
 $n/3213876088517980551083924184682325205044405987565585670602752$
 $n/6427752177035961102167848369364650410088811975131171341205504$
 $n/12855504354071922204335696738729300820177623950262342682411008$
 $n/25711008708143844408671393477458601640355247900524685364822016$
 $n/51422017416287688817342786954917203280710495801049370729644032$
 $n/102844034832575377634685573909834406561420991602098741459288064$
 $n/205688069665150755269371147819668813122841983204197482918576128$
 $n/411376139330301510538742295639337626245683966408394965837152256$
 $n/822752278660603021077484591278675252491367932816789931674304512$
 $n/1645504557321206042154969182557350504982735865633579863348609024$
 $n/3291009114642412084309938365114701009965471731267159726697218048$
 $n/6582018229284824168619876730229402019930943462534319453394436096$
 $n/13164036458569648337239753460458804039861886925068638906788872192$
 $n/26328072917139296674479506920917608079723773850137277813577744384$
 $n/52656145834278593348959013841835216159447547700274555627155488768$
 $n/105312291668557186697918027683670432318895095400549111254310977536$
 $n/210624583337114373395836055367340864637790190801098222508621955072$
 $n/421249166674228746791672110734681729275580381602196445017243910144$
 $n/842498333348457493583344221469363458551160763204392890034487820288$
 $n/1684996666696914987166688442938726917102321526408785780068975640576$
 $n/3369993333393829974333376885877453834204643052817571560137951281152$
 $n/6739986666787659948666753771754907668409286105635143120275902562304$
 $n/13479973333575319897333507543509815336818572211270286240551805124608$
 $n/26959946667150639794667015087019630673637144422540572481103610249216$
 $n/53919893334301279589334030174039261347274288845081144962207220498432$
 $n/107839786668602559178668060348078522694548577690162289924414440996864$
 $n/215679573337205118357336120696157045389097155380324579848828881993728$
 $n/431359146674410236714672241392314090778194310760649159697657763987456$
 $n/862718293348820473429344482784628181556388621521298319395315527974912$
 $n/1725436586697640946858688965569256363112777243042596638790631055949824$
 $n/3450873173395281893717377931138512726225554486085193277581262111899648$
 $n/6901746346790563787434755862277025452451108972170386555162524223799296$
 $n/13803492693581127574869511724554050904902217944340773110325048447598592$
 $n/27606985387162255149739023449108101809804435888681546220650096895197184$
 $n/55213970774324510299478046898216203619608871777363092441300193790394368$
 $n/110427941548649020598956093796432407239217743554726184882600387580788736$
 $n/220855883097298041197912187592864814478435487109452369765200775161577472$
 $n/441711766194596082395824375185729628956870974218904739530401550323154944$
 $n/883423532389192164791648750371459257913741948437809479060803100646309888$
 $n/1766847064778384329583297500742918515827483896875618958121606201292619776$
 $n/3533694129556768659166595001485837031654967793751237916243212402585239552$
 $n/7067388259113537318333190002971674063309935587502475832486424805170479104$
 $n/14134776518227074636666380005943348126619871175004951664972849610340958208$
 $n/28269553036454149273332760011886696253239742350009903329945699220681916416$
 $n/56539106072908298546665520023773392506479484700019806659891398441363832832$
 $n/113078212145816597093331040047546785012958969400039613319782796882727665664$
 $n/22615642429163319418666208009509357002591793880007922663956559376545533132$

RT analysis: Counting sortSort $A[1..n]$ with integer field "key" in range $1..k$:

```

1: for  $j \leftarrow 1$  to  $k$  do
2:    $C[j] \leftarrow 0$ 
3: for  $i \leftarrow 1$  to  $n$  do
4:    $C[A[i].key]++$ 
5: for  $j \leftarrow 2$  to  $k$  do
6:    $C[j] \leftarrow C[j] + C[j-1]$ 
7: for  $i \leftarrow n$  downto 1 do
8:    $j \leftarrow A[i].key$ 
9:    $B[C[j] - 1] \leftarrow A[i]$ 
10: return  $B$ 

```

$n \sim n$
 $? \text{ key } = k$

$2k$ [1
 1

$3n$ [1
 2

$3(k-1)$ [1
 2

$n \cdot S$ [1
 2
 2

$O(k+n)$

?
 Value of k ?

RT analysis: Rod-cutting [(0 - ∞)-knapsack] problem

```

1:  $R[0] \leftarrow 0$ 
2: for  $j \leftarrow 1$  to  $n$  do
3:    $q \leftarrow P[j]$ 
4:    $S[j] \leftarrow j$ 
5:   for  $i \leftarrow 1$  to  $j-1$  do
6:     if  $P[i] + R[j-i] > q$  then
7:        $q \leftarrow P[i] + R[j-i]; S[j] \leftarrow i$ 
8:    $R[j] \leftarrow q$ 
9: return  $R, S$ 

```

$\sum_{j=1}^n 2 + S(j-1)$
 $(j-1)S$ [2
 3
 1
 $\emptyset \ 2n + S \frac{n(n-1)}{2} - Sn$

$O(n^2)$

RT analysis: Activity selection problem

1: $ASP[0] \leftarrow 0$
 2: **for** $i \leftarrow 1$ **to** n **do**
 3: $j \leftarrow i - 1$
 4: **while** $f_j > s_i$ **do**
 5: $j \leftarrow j - 1$
 6: **if** $ASP[i - 1] \geq ASP[j] + P_i$ **then**
 7: $ASP[i] \leftarrow ASP[i - 1]; \text{ Use}[i] \leftarrow 'N'$
 8: **else**
 9: $ASP[i] \leftarrow ASP[j] + P_i; \text{ Use}[i] \leftarrow 'Y'; \text{ Pre}[i] \leftarrow j$
 10: **return** $ASP, \text{Use}, \text{Pre}$

RT analysis: Longest common subsequence problem

1: **for** $i \leftarrow 0$ **to** m **do**
 2: $C[i, 0] \leftarrow 0$
 3: **for** $j \leftarrow 0$ **to** n **do**
 4: $C[0, j] \leftarrow 0$
 5: **for** $i \leftarrow 1$ **to** m **do**
 6: **for** $j \leftarrow 1$ **to** n **do**
 7: **if** $x[i] = y[j]$ **then**
 8: $C[i, j] \leftarrow C[i - 1, j - 1] + 1; \text{ B}[i, j] \leftarrow 'D'$
 9: **else if** $C[i, j - 1] > C[i - 1, j]$ **then**
 10: $C[i, j] \leftarrow C[i, j - 1]; \text{ B}[i, j] \leftarrow 'L'$
 11: **else**
 12: $C[i, j] \leftarrow C[i - 1, j]; \text{ B}[i, j] \leftarrow 'U'$
 13: **return** C, B

m = first length n = second length

RT analysis: Output an actual LCS

```

1:  $i \leftarrow m$ 
2:  $j \leftarrow n$ 
3:  $k \leftarrow C[m, n]$ 
4: while  $k > 0$  do
5:   if  $B[i, j] = 'D'$  then
6:      $z[k] \leftarrow x[i]$ 
7:      $k \leftarrow k - 1$  ;  $i \leftarrow i - 1$  ;  $j \leftarrow j - 1$ 
8:   else if  $B[i, j] = 'L'$  then
9:      $j \leftarrow j - 1$ 
10:  else
11:     $i \leftarrow i - 1$ 
12: return  $z$ 

```

$k = C[m, n] = \text{Value at}$

RT analysis: Huffman coding

Input: Alphabet Σ , frequency of each character. Let $n = |\Sigma|$

1. For each character $c \in \Sigma$, create a tree with a single node containing element c , with priority $f(c)$
2. Create a priority queue with these n trees
3. While the priority queue has more than one tree, remove two least frequent trees x and y , merge them together, and insert the resulting tree into the priority queue, with priority $f(x) + f(y)$
4. Return the single tree in the priority queue

RT analysis: APSP by extension

1: **for** $u \in V$ **do**
 2: **for** $v \in V$ **do**
 3: $D[u, v] \leftarrow w(u, v)$
 4: $k \leftarrow 1$
 5: **while** $k < |V| - 1$ **do**
 6: **for** $u \in V$ **do**
 7: **for** $v \in V$ **do**
 8: $T[u, v] \leftarrow D[u, v]$
 9: **for** $m \in V$ **do**
 10: **if** $T[u, v] > D[u, m] + D[m, v]$ **then**
 11: $T[u, v] \leftarrow D[u, m] + D[m, v]$
 12: $D \leftarrow T$
 13: $k \leftarrow 2 * k$
 14: **return** D

RT analysis: Floyd-Warshall's algorithm for APSP

1: **for** $u \in V$ **do**
 2: **for** $v \in V$ **do**
 3: $D[u, v] \leftarrow w(u, v)$
 4: **for** $k \leftarrow 1$ **to** $|V|$ **do**
 5: **for** $u \in V$ **do**
 6: **for** $v \in V$ **do**
 7: $D[u, v] \leftarrow \min(D[u, v], D[u, k] + D[k, v])$

RT analysis: KMP algorithm for string matching

1: $\pi[1] \leftarrow 0$
 2: $k \leftarrow 0$
 3: **for** $q \leftarrow 2$ **to** m **do**
 4: **while** $k > 0$ **and** $P[k+1] \neq P[q]$ **do**
 5: $k \leftarrow \pi[k]$
 6: **if** $P[k+1] = P[q]$ **then**
 7: $k \leftarrow k+1$
 8: $\pi[q] \leftarrow k$
 9: **return** π

$\pi[k] = ?$ index

$$0 < \pi[k] < k$$

RT analysis: Counting sort

for($q=1$; $q \leq k$; $q++$) {
 $L[q] = \text{new LinkedList}();$
 }
 for($i=1$; $i \leq n$; $i++$) {
 $L[A[i].\text{key}].\text{add}(A[i]);$
 }
 $k = 1;$
 for($q=1$; $q \leq k$; $q++$) {
 for($x: L[q]$) {
 $A[k] = x;$
 $k++;$
 }
 }

$O(k)$

$O(n)$

$O(k^2)$

$O(k)$

$O(k^2 + n)$