Algoritmus Re--Pair, poprvé představen v článku \cite{892708}, je kompresní algoritmus založený na bezkontextové gramatice. Na vstupu dostává tento algoritmus řetězec znaků, např. text, který převede na řetězec terminálních a neterminálních symbolů bezkontextové gramatiky. Autoři zařadili tento algoritmus do skupiny \emph{off-line} slovníkových kompresních metod.

\emph{Off-line} metody využívají celého vstupního řetězce k vytvoření slovníku. Tyto metody tedy musí načíst celý vstupní soubor nebo jeho část do paměti. Části se stále rozumí mnohem větší část, než využívají \emph{on-line} metody. Algoritmy v \emph{on-line} skupině, tvoří slovník pomocí té části vstupu, kterou již přečetli, jak je tomu například u slovníkových kompresí LZ. Nevýhodou off-line metod je tedy velká paměťová náročnost. Oproti tomu výhodou, důvodem, proč vůbec uvažovat o této skupině, je možnost vytvořit slovníkové fráze, vedoucí k lepším kompresním výsledkům. Díky znalosti vstupních dat jako celku známe obecně lepší souvislosti.

V tomto dokumentu budeme popisovat kompresi textu a očekáváme od čtenáře základní znalost bezkontextové gramatiky. Algoritmy využívají bezkontextové gramatiky převádějí vstupní text na sekvenci terminálních a neterminálních symbolů. Aby mohli provést tuto transformaci musí nalézt přepisovací pravidla, které budou vést k maximální kompresi. Ukázku, jak můžeme transformovat text \texttt{abrakadabra} pomocí bezkontextové gramatiky nalezneme v Tabulce \ref{tab:cfg}. Záměrně jsme využili pravidla, které mají na pravé straně pouze dva symboly.

Je tedy zřejmé, že abeceda vzniklá na konci komprese, bude oproti terminálním znakům z textu, obsahovat také nové neterminální symboly. Výsledkem algoritmu je tedy řetězec a množina přepisovacích pravidel. Obě tyto informace musí být uloženy v komprimovaném souboru, nebo přeneseny přes síť. K zakódování se většinou využívá různých entropických metod, generující kódy proměnlivé délky, jako je například Huffmanovo kódování nebo Aritmetické kódování.

Samotná název Re--Pair je zkratkou anglického \emph{Recursive Pairing}, který už sám o sobě napovídá, že se bude jednat o metodu využívající určité rekurze a párování. Originální algoritmus vždy nahrazuje nejčetnější pár v textu. Párem rozumíme dvojici terminálních nebo neterminálních symbolů. Označíme-li pomocí malých písmen terminální symboly a pomocí velkých písmen neterminální symboly, tak mohou nastat tři situace:

\noindent Algoritmus nijak neupřednostňuje žádnou z nich. Nahrazování probíhá do té doby, dokud se v nahrazeném textu nachází určitý pár alespoň dvakrát. Postupně upravovaný text je dobrý způsob, jak si představit fungování algoritmu, ale při samotném běhu není žádné nahrazování prováděno. Dále bude popsáno, jak dokáže algoritmus postupně redukovat text a přitom vědět, v jakém stavu se text zrovna nachází.

Neterminál + Nete

On-line kompresní model – Slovník je postupně inkrementálně tvořen z frází, které jsou mu k dispozici z přečtené části textu. Slovník mže být měněn po každé frázi.

Off-line kompresní model – Slovník je tvořen pomocí celého textu nebo podle jeho určité části. Výhodou je lepší možnost vybrat správné fráze do slovníku, které maximalizují kompresi.

Re-Pair je schéma tohoto off-line kompresního modelu. Nabízí rychlou dekompresi, která se hodí, když chceme něco hledat v komprimovaném textu.

Pouze tvorba slovníku je off-line s potřebou mít celý text nebo část v paměti.

Symbolem označujeme jak symbol textu, tak i speciální symboly gramatiky. Symboly textu jsou označovány jako znaky. V textu je celkem *k* unikátních znaků, ale mnohem více *k’* symbolů. Délka textu je *n*.

Využívají hierarchického schématu, kde dlouhé fráze jsou kódovány pomocí odkazů na kratší fráze. Každá fráze v Re-Pair je využita k buď k přímému zakódování dvou části nebo jako stavební blok delší fráze, která je použita alespoň dvakrát.

Během derivace frází, jsou nejčetnější páry (dvojice symbolů) nahrazovány novým symbolem, reprezentující pár. Tímto vznikne nová abeceda a četnost páru musí být přepočtena. Toto se opakuje, dokud existuje nějaký pár, který se nachází alespoň dvakrát.

*Very simple pseudo code*

Po provedení tohoto algoritmu je zpráva redukována na sekvencí symbolů. Každý symbol reprezentuje znak nebo pár rekurzivních symbolů. Dá se představit jako strom, kde v listech se nacházejí znaky. Posledním krokem komprese entropické kódování symbolů, Huffman nebo Aritmetické kódování.

Může nastat situace, kdy existuje více jak jeden pár s maximální četností, ale volba, který pár bude nahrazen první má minimální vliv na výsledek. Přesto je vybrán pár, který je zatím nejmíň krát nahrazen.

*Ukázka nahrazení*

Popis implementace v lineárním čase *O(n)* a prostoru

Tři klíčové datové struktury

1. Pole symbolů – Je v něm uložena sekvence reprezentující symboly. Na začátku je v tomto poli informace o vstupní zprávě, ASCII znacích. Každý prvek pole obsahuje 3 čísla.
   1. CODE – Jedno číslo reprezentuje „číslo“ symbolu
   2. next – ukazatel na předcházející pozici v textu, ze které můžeme přečíst stejný pár symbolů
   3. prev – ukazatel na nadcházející pozici v textu, ze které můžeme přečíst stejný pár symbolů
2. Prioritní fronta – je implementována jako pole cca sqrt(n) seznamů. V prvním seznamu, nahoře, jsou páry s maximální četností. Pod tímto seznamem, dále v poli, jsou seznamy, kde vždy jsou páry s menší četností.
   1. Priorita je dána četností párů symbolů
   2. Po každém přidání nového páru do fronty je celá fronta aktualizována.
3. Hashovací tabulka – Obsahuje aktivní páry, které mohou být využity

Jelikož hashovací tabulka i prioritní fronta využívají stejné informace o „páru“, tak je tento pár implementován pomocí jedné struktury Pair. Tato struktura obsahuje:

* h\_next – ukazatel na další Pair se stejným hashem
  + hash je pár symbolů nebo znaků
* p\_prev – ukazatel na předcházející Pair se stejnou četností
* p\_next – ukazatel na následující Pair se stejnou četností
* f\_pos – pozice prvního výskytu páru ve vstupní zprávě
* b\_pos – pozice posledního výskytu páru ve vstupní zprávě
* left – levý symbol
* right – pravý symbol
* freq – četnost výskytu Pair

Během nahrazování párů jsou některé pozice v originálním textu *w* prázdné, a právě proto používáme ukazatele next a prev.

Můžeme se všimnout, že četnost existujících aktivních páru nikdy neroste, spíše klesá, když je jeden z jeho symbolů pohlcen některým nahrazeným párem. Při zmenšení četnosti dojde k přesunu do méně prioritního seznamu. Zároveň četnosti nových párů nejsou nikdy větší než četnosti nahrazených párů.

Nakonec jsou páru pouze v poslední listu fronty, který je lineárně proskenován a páry v něm jsou nahrazeny.

Fronta je inicializována lineárním skenem vstupní zprávy, četnosti jsou aktualizovány přes vyhledání v hashovací tabulce v čase O(n).

Jelikož při každé náhradě páro dochází k redukci délky textu je počet nahrazení O(n). Nahrazení nejčetnějšího páru je provedeno následovně:

1. Ve zprávě je nalezen první nebo další výskyt páru *ab* a jsou identifikovány sousedící symboly *x* a *y*, takže máme text *xaby*
2. Snížíme prioritu, četnost párů *xa* a *by*
3. Pár *ab* nahradíme symbolem *A*, dostaneme *xAy*
4. Zvýšíme prioritu nebo přidáme nové páry *xA* a *Ay*

Při implementaci se musí dát pozor na nahrazení sekvencí stejných symbolů, například *aaaa* musí být nahrazeno jen 2 symboly a ne třemi.

Operace 1 a 3 jsou provedeny v konstantním čase pomocí ukazatelů next a prev

Snížení a zvýšení priority je také provedeno v konstantním čase, neboť Pair obsahuje ukazatel na prioritní fronty. Fronty jsou prolinkovány a tudíž je přesun mezi nimi velmi lehký.

Různé varianty:

* Phrase Browsing – pracuje na bázi celých slov namísto jednotlivých znaků
  + <https://github.com/rwanwork/Re-Pair>
* Original
* Better encoding – Lepší kódování pravidel pomocí kódu proměnné délky
  + <https://github.com/syoshid/Re-Pair-VF>
* Better memory usage – Algoritmus funguje ve dvou fázích, high-frequency a low-frequency
  + V první fází jsou nahrazovány páry s velkou četností
  + V druhé fází jsou nahrazovány páry s menší četností
  + Obě fáze se liší frontou, kterou využívají k vybírání páru k nahrazení
  + <https://github.com/nicolaprezza/Re-Pair>
  + Zkoušeno
* MaximalRepeat-Repair
  + <https://github.com/tkida/MR-Repair>
  + Menší velikost vygenerované gramatiky
  + Když je pár identifikován jako nejčastější, tak se snaží najít řetězec nejdelší možné délky
  + Nedokončeno